**Ryan Timbrook**

**David Madsen**

**Diego Vales**

**Course:** IST 736 Text Mining

**Term:** Fall, 2019

Table of Contents

[1 Introduction 3](#_Toc26446243)

[1.1 Purpose 3](#_Toc26446244)

[1.2 Scope 3](#_Toc26446245)

[2 Initial Data Mining 4](#_Toc26446246)

[2.1 About the Data 4](#_Toc26446247)

[2.1.1 Dataset Info 4](#_Toc26446248)

[2.1.2 Data Exploration & Cleaning 4](#_Toc26446249)

[3 Sentiment Classification Modeling 7](#_Toc26446250)

[3.1 Analysis and Models 7](#_Toc26446251)

[NFL Teams - 7](#_Toc26446252)

[3.1.1 Data Transformation and Cleaning 7](#_Toc26446253)

[3.1.2 Models 7](#_Toc26446254)

[4 Topic Modeling 8](#_Toc26446255)

[4.1 Analysis and Models 8](#_Toc26446256)

[NFL Teams - 8](#_Toc26446257)

[4.1.1 Data Transformation and Cleaning 8](#_Toc26446258)

[4.1.2 Models 8](#_Toc26446259)

[5 Player Stats Outcome Prediction 9](#_Toc26446260)

[5.1 Analysis and Models 9](#_Toc26446261)

[NFL Teams - 9](#_Toc26446262)

[5.1.1 Data Transformation and Cleaning 9](#_Toc26446263)

[5.1.2 Models 9](#_Toc26446264)

[6 Conclusion 10](#_Toc26446265)

[7 Appendix: References 11](#_Toc26446266)

# Introduction

The NFL was already a lucrative business before Fantasy football took off, now you can't turn on a news program without hearing about Fantascy stats and which provider has better predictive modeling and real-time feedback on player's performance. According to Gallup, Football still overwhelmingly dominates American enthusiasm, with 37% calling it their favorite sport, compared to 11% for basketball, it's the closest rival; Baseball is at 9%, after decades of declining stature. NFL revenue grew an estimated $900 million to $14 billion in 2017, in 2018 it generated about $15 billion. The league announced in January that it's aiming to boost it's annual revenue to $25 billion by 2027. Fantasy football and the spread of legalized sports betting across the U.S. promises to lock in fans and keep them focused on the game.

Data Science aims to be at the heart of how fantasy players and those gambling on teams make their choices. IBM announced this 2019 season, "Fantasy Insights with Watson". "ESPN Fantasy Insights draws upon the latest in machine learning techniques to turn unstructured data into valuable insights. Nearly 10 million players rely on the combined resources of Watson Discovery and Watson OpenScale running on the IBM Cloud to give them a competitive edge."

*- https://ibm.com/fantasy*

In this research the aim is to use text mining techniques on public news and media web sites to aggregate data relating to NFL teams and it's players to determine if sentiment opinion alone is a predictor of Player Fantacy Football performance stats.

An example would be, if Deshaun Watson's Fantacy Football stat forecast for week-6 is 85 pts leading into that weeks game, can the public opinion for that week of negative, neutral or positive predict if he will achieve that points threshold or not. Thereby giving a player of Fantacy Football an edge in choosing to either keep Deshaun Watson as his/her starting quarterback or bench him for someone else.

## Purpose

Identify public sentiment toward NFL teams and it's players that could help fans choose teams and players to play in their fantasy leagues.

## Scope

* Gather public data using text mining techniques on:
  + Public opinion toward NFL teams, coaches, and players.
    - Reduce data gathering to one of each for initial POC.
    - Data is in timeseries format from the first week of the NFL 2019 session to current schedule week
  + Players weekly Fantasy Football performance stat forecasts on a daily time scale.
* Perform sentiment analysis modeling ML techniques on data set to determin model prediction accuracies.
* Perform unsuppervised topic modeling ML techniques on document text to gain insights into public opinion primary opinion drivers.
* Evaluate weekly sentimnent trends aligning with Fantascy Football performance stat forecasts.

# Initial Data Mining

## About the Data

Gathering data on the NFL Team, Coach and Player selected for this experiment was mined through the Twitter Developer Platform, Twitter APIs. In order to have data for analysis which covered the entire 2019 NFL weekly schedule, multiple Twitter API types were required.

Additional details on each of the API capabilities and endpoints can be found in table 2.1.1 below.

### Search Tweets Features Used

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Category** | **Product name** | **Supported history** | **Query capability** | **Counts endpoint** | **Data fidelity** |
| **Standard** | [Standard Search API](https://developer.twitter.com/en/docs/tweets/search/overview/standard) | 7 days | [Standard operators](https://developer.twitter.com/en/docs/tweets/search/guides/standard-operators) | Not available | Incomplete |
| **Premium** | [Search Tweets: 30-day endpoint](https://developer.twitter.com/en/docs/tweets/search/overview/premium) | 30 days | [Premium operators](https://developer.twitter.com/en/docs/tweets/search/guides/premium-operators) | Available | Full |
| **Premium** | [Search Tweets: Full-archive endpoint](https://developer.twitter.com/en/docs/tweets/search/overview/premium) | Tweets from as early as 2006 | [Premium operators](https://developer.twitter.com/en/docs/tweets/search/guides/premium-operators) | Available | Full |

### Tweet Objects

Tweets are the basic atomic building block of all things Twitter. Tweets are also known as “status updates.” The Tweet object has a long list of ‘root-level’ attributes, including fundamental attributes such as id, created\_at, and text. Tweet objects are also the ‘parent’ object to several child objects. Tweet child objects include user, entities, and extended\_entities. Tweets that are geo-tagged will have a place child object

When mining for tweets, tweet objects are returned as JSON objects similar to this example structure:

The JSON will be a mix of ‘root-level’ attributes (here we are highlighting some of the most fundamental attributes), and child objects (which are represented here with the {} notation):

{

"created\_at": "Wed Oct 10 20:19:24 +0000 2018",

"id": 1050118621198921728,

"id\_str": "1050118621198921728",

"text": "To make room for more expression, we will now count all emojis as equal—including those with gender‍‍‍ ‍‍and skin t… https://t.co/MkGjXf9aXm",

"user": {},

"entities": {}

}

\*\*Tweet Data Dictionary can be found [here](https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/tweet-object).

### Premium Search Basic Requirements

The following is needed:

* [A developer account](https://developer.twitter.com/en/dashboard)
* [A registered app](https://developer.twitter.com/en/apps)
* [A developer environment setup](https://developer.twitter.com/en/account/environments)
* Authentication - for cURL need a [bearer token](https://developer.twitter.com/en/docs/basics/authentication/guides/bearer-tokens), for Twurl need to have [Twurl setup](https://developer.twitter.com/en/docs/tutorials/using-twurl.html)

### Accessing the data endpoint

The data endpoint will provide the full Tweet payload of matched tweets. Use the from: and lang: operators to find Tweets originating from @TwitterDev in English. *For more operators*[*click here*](https://developer.twitter.com/en/docs/tweets/search/guides/premium-operators.html)*.*

Details on the data endpoint response payload that is returned from the API search request can be found [here](https://developer.twitter.com/en/docs/tweets/search/quick-start/premium-30-day).

#### Search Endpoints

* **30day**: "https://api.twitter.com/1.1/tweets/search/30day/sandbox.json"
  + Requests Usage Monthly limit: 250
  + Rate Limit Per Request: 100
* **fullarchive**: "https://api.twitter.com/1.1/tweets/search/fullarchive/devfullarchive.json"
  + Requests Usage Montly limit: 100
  + Rate Limit Per Request: 500

#### Search Parameters

* query:
  + These were the individual search terms used based on nfl type being processed
    - Team: "houston texans"
    - Coach: "bill obrian"
    - Player: "deshaun watson"
* **fromDate**: shown below in section 2.1.4.2.1
* **toDate**: shown below in section 2.1.4.2.1
* **maxResults**: shown below in section 2.1.4.2.1
* **next\_page**: this value is returned in the JSON response payload per request

##### Search Date Range Endpoint Mapping

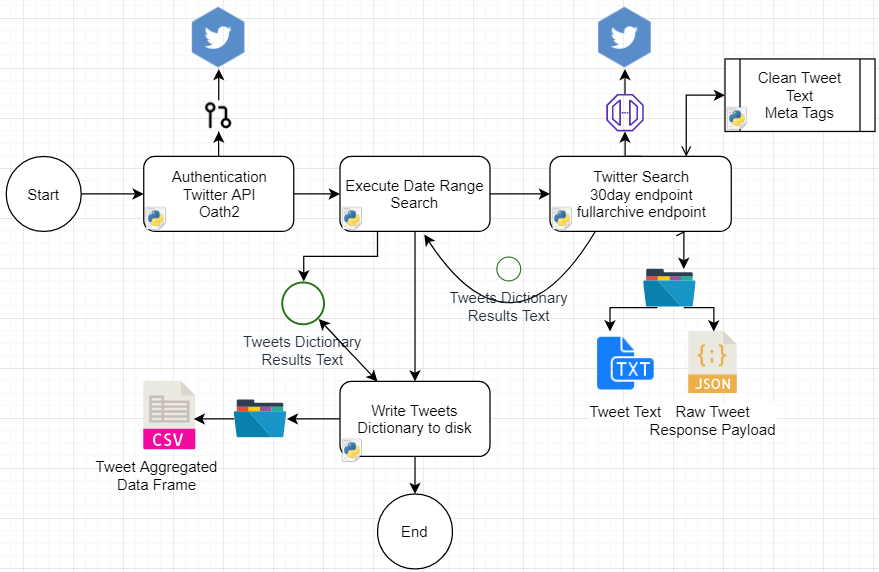
For each NFL type, team, coach, and player, individual search requests were performed specific to a week date range which represents the NFL game schedule. For this, different search API endpoints were utilized based on their search time frame capabilities and rate limit restrictions.

|  |  |  |  |
| --- | --- | --- | --- |
| NFL Weekly Schedule | From Date - To Date | Search Endoint | Max Results |
| Week 1 | ('201909010000','201909080000') | fullarchive | 500 |
| Week 2 | ('201909080000','201909150000') | fullarchive | 500 |
| Week 3 | ('201909150000','201909220000') | fullarchive | 500 |
| Week 4 | ('201909220000','201909290000') | fullarchive | 500 |
| Week 5 | ('201909290000','201910060000') | fullarchive | 500 |
| Week 6 | ('201910060000','201910130000') | fullarchive | 500 |
| Week 7 | ('201910130000','201910200000') | fullarchive | 500 |
| Week 8 | ('201910200000','201910270000') | fullarchive | 500 |
| Week 9 | ('201910270000','201911030000') | fullarchive | 500 |
| Week 10 | ('201911030000','201911100000') | 30day | 100 |
| Week 11 | ('201911100000','201911170000') | 30day | 100 |
| Week 12 | ('201911100000','201911170000') | 30day |  |
| Week 13 | ('201911240000','201912010000') | 30day |  |

### Twitter Search Function Behavior

Jupyter Notebook Names:

* **search\_twitter\_nfl\_coach\_premium**.ipynb
* **search\_twitter\_nfl\_team\_premium**.ipynb
* **search\_twitter\_nfl\_player\_premium**.ipynb



#### Data Cleaning

At this stage, tweet text was cleaned of hashtags, urls, and @tags. Each of these types were captured and stored into lists. During the final stage of the process after all tweets returned from all requests, these data elements were packaged into a dataframe and written to disk as a csv file. Each of the records has an ID attribute that maps it back to the original tweet document it's associated to.

#### Data Modeling

The tweet data collected during the search process captures

* **id**: tweet object response attributed (result["id\_str"])
* **created\_at**: tweet object response attribute (result["created\_at"])
* **date**: aggregated composite value created from created\_at value
* **time**: aggregated composite value created from created\_at value
* **user**: tweet object response attributed (result["user"]["screen\_name"])
* **text**: tweet object response attributed (result["text"])
* **favorite\_count**: tweet object response attributed (result["user"]["favourites\_count"])
* **year**: aggregated composite value created from created\_at value
* **month**: aggregated composite value created from created\_at value
* **day\_of\_month**: aggregated composite value created from created\_at value
* **day\_of\_week**: aggregated composite value created from created\_at value

#### Data Export

Four data files are generated as output from this process.

These first two are written out at the end of the process from memory DataFrame objects that were updated throughout the search iterations. It contains a complete list of all tweet documents collected.

Output OS Path Patterns: outputPath = f'{dataDir}/{nfl\_type}/{search\_on}/v{search\_iteration}'

* search\_result\_tweet\_text\_data.csv
* search\_result\_tweet\_text\_meta.csv

This process, when executed over the three NFL types, outputs three csv data files:

* **coach\_search\_results\_tweet\_data.csv**
* **player\_search\_resutls\_tweet\_data.csv**
* **team\_search\_results\_tweet\_data.csv**

These two files are written to during a search process where the process iterates over a list of tweets returned in each request.

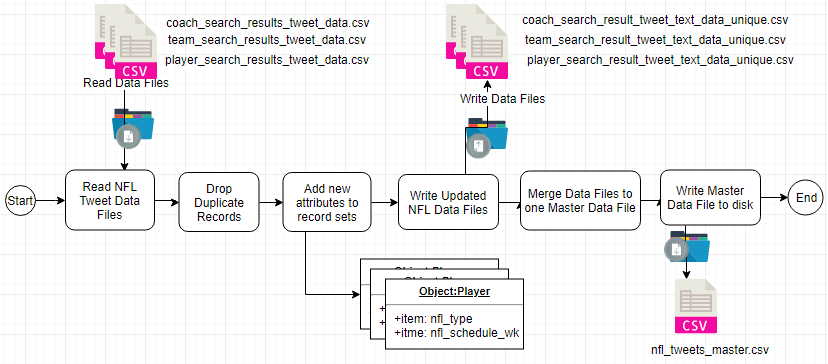
Output OS Path Patterns: outputPath = f'{dataDir}/{nfl\_type}/{search\_on}/v{search\_iteration}/{search\_range}'

* tweet\_filename=f'{outputPath}/tweet\_text.txt'
* raw\_filename=f'{outputPath}/tweet\_raw.txt'

### Merge NFL Data Sets to Master File

Jupyter Notebook: **merge\_datasets\_to\_master**.ipynb

This function of the data engineering step reads in the three separate NFL Type data sets and merges them into one data set for model training and analysis. Two additonal categorical attributes were added to the data sets during this process. The first was a field called nfl\_type. The values are 'coach','team','player'. It's used to segment the data for post processes grainular analysis. The other field is a numeric categorical attribute that identifies the nfl schedule week the tweets are associated to from a timeline perspective.



#### Data Export

Four data files are generated as output from this proces.

* nfl\_tweets\_master.csv
* coach\_search\_result\_tweet\_text\_data\_unique.csv
* team\_search\_result\_tweet\_text\_data\_unique.csv
* player\_search\_result\_tweet\_text\_data\_unique.csv

### VEDAR Sentiment Classification

Jupyter Notebook: classify\_train\_nfl\_master.ipynb

##### Vectorization Preprocessing Steps

For each text document, the following pre-processing vectorization steps were taken:

(note - each of the bellow steps is controlled via a Boolean True or False conditional statement that allowed the testing of each of these steps independently as well as in combination to evaluate optimal vectorization preparation)

* + All hashtag tokens were removed
  + All URL tokens were removed
  + Punctuation was removed using the python string. punctuation values
    1. '!"#$%&\'()\*+,-./:;<=>?@[\\]^\_`{|}~'
  + Non-Alphabetic tokens were removed using the python string method isalpha()
  + Lowercase all of the token characters
  + Stop words were removed using the NLTK English stopwords list
    1. additionally, this step allows the addition of custom stop words to be added to the list for fine-tuning.

Post-pre-processing: Each cleaned text was saved to its own file to be used as a corpus of documents in the vectorization process.

Total Feature Count Prior to Vectorization Preprocessing:

Table 2.3: Vocabulary Size Reduction Comparisons (small sample)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***After Preprocessing***  ***Feature Count*** | **Remove Stop Word** | **Remove Punctuation** | **Remove Non-Alpha** | **Lowercase** | **Stemming** |
|  | True | True | True | True | False |
|  | True | True | True | True | True |
|  | True | True | True | False | False |
|  | True | True | False | False | False |
|  | False | False | True | False | True |
|  |  |  |  |  |  |

As shown above in Table 2.3, each of the cleaning steps affects the total feature set vocabulary in varying ways. Ideally, given the time, it's recommended to save a new file after each transformation to evaluate the linguistical impacts it has on the overall quality of the new dataset in performing sentiment analysis on the tweet texts.

One example of needing to run many scenarios to pick the best preprocessing techniques for this task is the lowercasing of the dataset. Often people use capitalization to convey a tone and or strength to a statement they are making through text. By removing these human expressions of sentiment we're possibly missing out on valuable insights and could skew the data and it's meaning.

For our purposes of this trial the first configuration set shown above in Table 2.3 was used for the Vectorization modeling that will be described in section 3.

|  |  |
| --- | --- |
| Figure 2.4: Top 10 Cleaned Feature Counts | Image 2.3: Word Cloud of full corpus cleaned |

# Sentiment Classification Modeling

## Analysis and Models

### NFL Teams -

### Data Transformation and Cleaning

### Models

#### Model Details

#### Model Parameters

# Topic Modeling

## Analysis and Models

### NFL Teams -

### Data Transformation and Cleaning

### Models

#### Model Details

#### Model Parameters

#### Model Results

# Player Stats Outcome Prediction

## Analysis and Models

### NFL Teams -

### Data Transformation and Cleaning

### Models

#### Model Details

#### Model Parameters

#### Model Results

# Conclusion

# Appendix: References

1. Twitter Developer Portal - Product APIs: <https://developer.twitter.com/en/products/products-overview>
   1. Twitter Search Tweets Overview: <https://developer.twitter.com/en/docs/tweets/search/overview>
2. Test
3. Test2
4. Test3