
FINAL REPORT/DOCUMENTATION

Title: NFL Team Reddit Sentiment Analyzer

PROJECT PRESENTATION LINKS:

BOX: https://uofi.box.com/s/a16ls7nh8qwezx8l2k4f8s1d0z867yrg

GOOGLE DRIVE: https://drive.google.com/file/d/1j1UsosUmW6z-KpoA6Szv33gBcYFyaUwr/view?usp=sharing

README Outline:
This README document is split into the following sections.
1) Overview of the function of the code
2) Documentation of how the software is implemented
3) Detailed Instructions on how to install and run software
4) Evaluation of Results and Project

---

1) [Overview of Function of Code] Overview of the function of the code

This application uses the Python Reddit API Wrapper (PRAW) to perform sentiment analysis on the public opinion found in different NFL team subreddits. For a specified set of NFL teams, the corresponding subreddit is analyzed to retrieve text data for a specified number of posts. For each post in the NFL team's subreddit, the text is cleaned. Once cleaned, a sentiment score is calculated for each post using The Natural Language Toolkit's Sentiment Intensity Analyzer. Each of these scores is used to calculate an average sentiment score for the entire NFL team's subreddit (representing the sentiment of the public opinion for the team). For each subreddit, a collection of the combined words from each post is created. This is used to build a frequency distribution for the subreddit to determine what the most common words are. The sportradar API is used to retrieve current NFL statistics. This application uses the API to retrieve a team's win total, the outcome of their most recent game, and points for total (team's total amount of points scored) for each of the specified teams. These statistics are compared to the average sentiment score. Correlations are calculated and displayed for (wins/average sentiment), (most recent outcome/average sentiment), and (points for/average sentiment). Predictions for the outcome of each team's next game are made based off of the team's public sentiment score. If the team's average sentiment is greater than 0.3, then the predicted result of their next game is a win. If the team's average sentiment is less than 0.3, then the predicted result of their next game is a loss. The application also displays a dictionary containing wins, most recent outcome,

average sentiment, predicted result of next game, points for, and most common words for each specified team. Currently the application runs for a specific set of NFL teams specified in the main function. If additional teams need to be analyzed, the team name along with that team's subreddit name must be added to the lists at the top of the main function.

---

2) [Implementation Documentation] Documentation of how the software is implemented

def cleanText(text):
This function accepts a string as input and returns a cleaned string. Data cleaning techniques used include removing parenthesis, removing URLs, removing punctuation, and removing words that don't belong to the corpus.

def getStemListOfWords(dataDict):
This function accepts a dictionary as input and returns a dictionary as output after the values in the dictionary are stemmed. For every key-value pair in the input dictionary, the Lancaster Stemmer from the Natural Language Toolkit stems the value. This stemmed value is added to the output dictionary. This function isn't currently being used. It was used for analysis.

def removeStopwords(text):
This function accepts a list of words as input and returns the list after all stopwords are removed. The Natural Language Toolkits list of stopwords is used.

def getFreqDist(text):
This function accepts a string as input and returns a frequency distribution for the string as output. The Natural Language Toolkit's FreqDist() is used.

def getSubRedditPosts(team):
This function accepts a subreddit name for a specific NFL team as input and returns a dictionary of [post title, post comments] key-value pairs. PRAW retrieves the top posts from the specified subreddit, saving off the title and comments for each post. The comments are concatenated into one string to support a Bag of Words approach.

def getCollectionOfWords(dataDict):
This function accepts a dictionary as input and returns a string of words as output. The input dictionary has [post title, post comments] key-value pairs for a specific subreddit. The output string contains the comments for every post concatenated into a single string. This can later be used to determine the frequency distribution across the

entire subreddit instead of a single post.

def cleanSubRedditPosts(data):
This function accepts a dictionary as input and returns a dictionary as output. The input dictionary has un-edited [post title, post comments] key-value pairs for a specific subreddit. Data cleaning techniques edit the values for each key in the dictionary. The output dictionary contains comments for each post that no longer have stopwords, punctuation, etc.

def sentimentAnalyzer(data):
This function receives a dictionary as input and returns an average sentiment intensity score as output. The Natural Language Toolkit's pre-trained Sentiment Intensity Analyzer calculates intensity scores for each value in the [post title, post comments] key-value pair dictionary. The average intensity score across all posts for the specific subreddit is calculated.

def parseJson(jsonfile, teamNames):
This function receives a path to a json file and a list of team names as input and returns a dictionary as output. The json file is retrieved using the sportradar API and contains statistics for each NFL team. The statistics that this application is interested in are total wins (denoted as "wins" in the json file) and outcome of the most recent game (denoted as "streak" in the json file). For each team in the team names input list, a dictionary entry is created to store the statistics. These statistics are later compared to the sentiment analysis results to determine if a correlation exists.

def getJson():
This function retrieves a json file from the sportradar api, saves the file, and returns the file.

def calculateCorrelation(dataDict):
This function receives a dictionary as output and prints two correlation values. The input dictionary contains key-value pairs of team-team statistics. For every team, the correlation between wins/ average sentiment and most recent outcome/average sentiment are calculated.

def plotData(dataDict):
This function receives a dictionary as input and plots a scatter plot. The scatter plot contains data points for every wins/average sentiment pair. This function isn't currently being used.

def getGamePrediction(averageSentiment):
This function receives an average sentiment value as input and returns a prediction for the result of the team's next game.

def getMostFavoredTeam(dataDict):

This function receives a dictionary as input and prints a team name. The team name printed will have the highest public sentiment.

def getLeastFavoredTeam(dataDict):
This function receives a dictionary as input and prints a team name. The team name printed will have the lowest public sentiment.

def main():
This function ties everything together. The following steps provide an overview.
1) Lists are created for the NFL teams that this application is going to provide sentiment analysis for. The first list contains formatted team names. The second list contains the applicable subreddit names for each team that don't follow the same format (some subreddits specify the city in addition to the team name ex. jets vs. nyjets)
2) Json file is parsed. A dictionary containing statistics for each team is returned.
For each team...
3) Posts are retrieved from the specified subreddit. This includes post title and post comments.
4) Data cleaning techniques are performed on the retrieved subreddit posts.
5) Sentiment Analysis is performed on each post from the subreddit. An average sentiment value is retrieved.
6) A collection of words is retrieved after combining every comment from every post in the specific subreddit.
7) A frequency distribution is calculated for the entire collection from this team's subreddit.
For each team ends...
8) The program outputs the dictionary containing the statistics for each team (wins, streak, most common words, average sentiment, points for, and predicted result of next game)
9) The program outputs the correlation between (wins vs average sentiment), (most recent outcome vs. average sentiment), and (points for vs. average sentiment)
10) The program outputs the team with the highest sentiment and the team with the lowest sentiment.

---

3) [Usage Documentation] Detailed Instructions on how to install and run software

Files needed to run application.
Download the contents of the repository and navigate to this directory on your computer. This application is contained within a single python file titled CourseProject/Main.py. In addition to this python file you'll also need the json file located at CourseProject/data/nfl.json. Please verify that your directory contains both the Main.py application file and the data/nfl.json json file.

Setting up your environment.
In order to download the necessary python modules to run this application you'll need to run the following set of commands.

```
> curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py

> python get-pip.py

> pip install praw

> pip install nltk

> pip install pprint (maybe not this one)

> pip install re (maybe not this one)

> pip install requests

> pip install json (maybe not this one)

> pip install numpy

> pip install scipy

> pip install matplotlib
```

Running the program.

```
> python Main.py
```

Program output.
In your console, you should see several different outputs after running the program. The title for each post will be displayed as the application retrieves data from reddit (this output isn't important). At the end, the application will output the dictionary of statistics and correlation scores. In each team's dictionary you'll find several statistics including the average sentiment and the predicted outcome of the next game.

---

4) Evaluation of Results and Project
According to the output of my application (when the data set is large enough), a team's win total is positively correlated to the sentiment of the public opinion, a team's most recent outcome (win/loss) is positively correlated to the sentiment of the public opinion, and a team's points for total (total amount of points scored in the season) is positively related to the sentiment of the public opinion. I expected a team's most recent outcome to have a high positive correlation with the sentiment of the public opinion. This is because

my application retrieves the most recent posts for each of the NFL team subreddits. For example, if a team recently lost, then its likely that fans will make negative posts related to the loss. I was surprised by the low positive correlation between a team's win total and the sentiment of the public opinion. I believe this is due to the most recent game having a large impact on a fans opinion. I also expected the positive correlation between points for and public sentiment. The goal of an NFL game is to score points, so if a team has a high points for total, that means they've been successful. This would lead to a positive public opinion

I was able to complete a majority of the application outlined in my original proposal. My application fetches text data from an online source, cleans the data, performs sentiment analysis on the data, makes predictions based off the average sentiment, and analyzes the correlation between a team's statistics and the public opinion for that team.

There were some parts of the initial proposal that I was unable to complete. Mainly, this includes analyzing the accuracy of my predictions. Currently, the application predicts the outcome of each team's next game. Since the next game has yet to occur, I can't determine the accuracy of my prediction. If I had more time, I could have tried to fetch old text data to make predictions on game results that have already occurred.

If I was given more time to work on this application I would have expanded several things. First off, I would have generated a data set of negative/positive opinions from reddit. This would have allowed me to train a model that could have been used to perform the sentiment analysis. I had a hard time finding a pre-existing data set that fir the criteria I was looking for. Also, I would have liked to expand this application to accept command line arguments from the user. The command line arguments would have included the sport the user wished to analyze along with the teams from that sport.