John Gordley

Professor David Chiang

CSE 40657 - Natural Language Processing

14 December 2021

(Please see project README.md for code information)

A ‘Wisdom of the Masses’ Approach to NFL Outcome Prediction

The goal of this project is to determine if Twitter conversation surrounding an NFL game in the week leading up to the game itself can provide any insight into the eventual winner of the game. This problem is usually tackled in the sports betting world in an attempt to win money by beating the sportsbooks in Vegas through better predicting methods using pregame numerical data and general statistics surrounding the game. I have always wondered whether or not there is some way to summarize the general feelings of each fan base to gain insight into how a team will perform on game day, so through this project I aimed to create a model that would take an upcoming game as input and search through Twitter to determine extra information other than simply numbers and stats for each team and make a prediction on the eventual winner. The methods used in this project may be helpful in a variety of contexts other than football prediction, for example other sports or even competitive events such as elections or fundraisers. In this report I will cover the data I used and how I collected it, the baseline metrics I used to measure success, and the results of the model that I implemented. The report concludes with an analysis of what I learned throughout this project and instructions on how to access my code and replicate my results.

During my preliminary research of previous attempts at using Twitter data to predict sporting outcomes, I came across work done by Kevin Gimpel in 2012 where he attempted to

augment current game data with Twitter data to predict an NFL game winner. Thankfully, he provided the dataset he used consisting of id's of 930,000 weekly tweets from the 2012 NFL season for others to use to try and tackle this problem. Weekly tweets are described as "those that occurred at least 12 hours after the start of the previous game and 1 hour before the start of the upcoming game for their assigned team", so they were exactly what I was looking for. To gather the text content of these Tweets, I signed up for a Twitter Developer Account and used the GET endpoint for multiple Tweets by id (which is limited to 100 ids per request and ~280 requests per 15 minute window). In my code to gather these tweets, I pulled tweets by id until I hit the 15 minute time limit, then used time.sleep() for 15 minutes and resumed tweet collection until all tweets were collected. Interestingly, as these tweets are almost 9 years old, many of them had been deleted or their accounts were no longer public and so the tweet content could not be gathered. 545705 out of 930000 tweets were public and available for an overall percentage of 58.7% that I was able to collect.

Although the Kevin Gimpel dataset included total points and winner of each game, I wanted to use more specific game information to make a prediction for each team. I built a web scraper (utilizing Beautiful Soup) to go to TeamRankings.com and gather passing-yards-per-game, points-per-game, rushing-yards-per-game, and turnover-margin-per-game for each team, for each week of the 2012 NFL season. In my prediction, I use the average over the previous 3 games for each prediction, and in the case of week 1 where there are no previous games, I use the previous season 2011 average for each statistic.

After collecting all of the game data and Tweet data that I needed to begin implementing my model, I came up with a baseline model for predicting a winner based on only game data.

The metric I used for success in this project was the overall correct prediction accuracy, as in the real world application for this model one can win money and be a successful sports bettor with a high enough accuracy. The baseline model I implemented was a KNN classifier with k=18 that achieved a test accuracy of 64.5%. This model was sufficient in my view to use as a baseline, as it achieved a relatively high accuracy, much better than randomly guessing a winner. With this baseline, I was ready to move on to creating a Tweet sentiment classifier.

To understand more about how to build a sentiment classifier to use on the tweets I collected, I completed the Coursera course Natural Language Processing with Classification and Vector Spaces by Younes Bensouda Mourri from Stanford. Here, I learned how to build the Naive Bayes model for classification that learns Tweet sentiment based on the NLTK twitter dataset. The model I created is very similar to the Naive Bayes model I implemented for the Dialogue Act Classifier assignment and performed very well on the test set with around 99.4% accuracy. The input to this classifier can simply be a Tweet string and the output is a number representing the sum of the sentiment of each word in the Tweet. A total sentiment > 0 is considered a 'positive' tweet and below 0 is a 'negative' Tweet.

For my sentiment score implementation, I tried out 2 different ways of generating a sentiment score for each team (home and away) before an upcoming game:

1. Aggregate method (sentiment_prediction.ipynb) - sum up the naive bayes predicted p value for each team to get a total score for each team
2. Proportion method (sentiment_prediction_proportion.ipynb) - tally the number of positive and negative tweets for each team and use the proportion of positive to negative tweets as the score

For both score methods, I tried out using only the home_sentiment and away_sentiment to predict a winner. Then, I moved on to combine the home_sentiment and away_sentiment scores with the rest of the previous game data I had collected to predict a winner. For reference, my baseline was 64.5% accurate with game data alone. The top methods Sentiment Aggregate and Sentiment Proportion represent the two scoring methods I used, and the first column represents using the sentiment scores by themselves to make a prediction (Standalone) and combining the sentiment scores with the previously collected game data (with Game Data). The resulting accuracies for each method combination is shown below in the Figure 1:

	Sentiment Aggregate	Sentiment Proportion
Standalone	0.644	0.555
with Game Data	0.708	0.688

Figure 1. Overall Accuracy by Model

The best K value I found by testing the KNN model for K values between 1 and 26. The results of these tests were then graphed as shown in Figure 2:

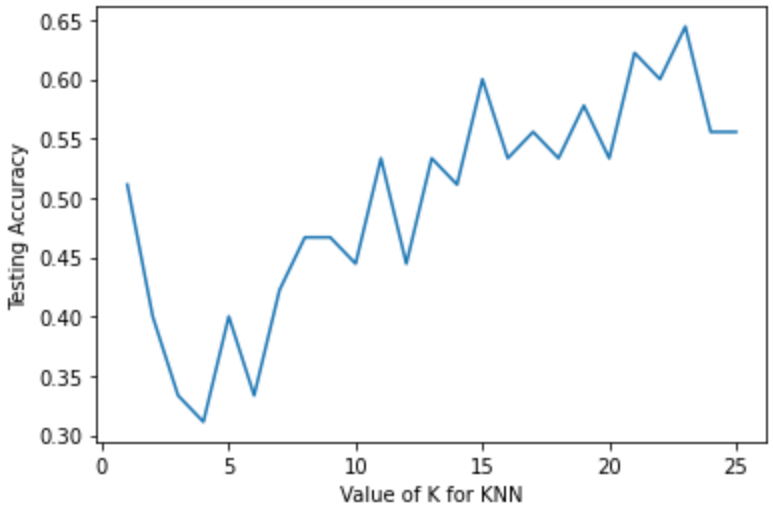

Figure 2. K value accuracies for Sentiment Aggregate Only

The optimal K values for each model are shown below in Figure 3:

	Sentiment Aggregate	Sentiment Proportion
Standalone	23	15
with Game Data	7	17

Figure 3. Optimal K Values for Each Model

As you can see from the accuracies of each model, each method of sentiment scoring on their own did not perform better than the baseline model using only pregame data. However, it is interesting to see that the aggregate score performed better than the proportion score. Most interesting are the results from combining the sentiment scores with pregame data, resulting in improved accuracies over the baseline model of only using game data. For the aggregate score, this increase over the baseline is a little over 5% which is a great improvement!

This project has been an incredible learning experience for me as it was my first time implementing a data science related project completely from scratch, my first time implementing

a naive bayes classifier, and first time using a web scraping tool to gather data. The data gathering and preparation were much more work than I expected but once I had all the data in place, it was extremely exciting to implement my different model choices! I would also like to offer up the data I collected to you in case you want to use it in your course at some point as it did take me quite a bit of effort to use and it is (hopefully) in a very easy to use format now. My main takeaway from this project was the power of probability based methods for sentiment classification. It is incredible how powerful the naive bayes model is in predicting sentiment and I really enjoyed implementing it from scratch. In the future, I would like to try and tackle this problem again and improve it in several ways, namely:

1. Using a positive/negative labeled Tweet dataset with more sport-specific vocabulary
2. Try and use an RNN classifier instead of a simple naive bayes classifier for sentiment
3. Use other, perhaps more powerful overall classification models for the final model instead of KNN

All of the code I wrote and data that I used can be found at the public github repo: https://github.com/jgordley/NLP-NFL-Prediction-Project. I have also uploaded my project as a zip file to Sakai. Please do not hesitate to reach out to me at my email jgordley@nd.edu with any questions! Thank you for a wonderful semester.