NBA Reddit Sentiment Analysis

Spring 2024

Andrew Branum
CSE 621
University of Louisville
Louisville, KY US
asbran03@louisville.edu

Nathaniel Cecil
CSE 621
University of Louisville
Louisville, KY US
ntceci01@louisville.edu

INTRODUCTION

The finale of the NBA season began less than two weeks ago. The eight best teams from the Western and Eastern Conferences will play several best-of-sevens in order to decide which team stands out above the rest as the best team in the country. It is a culmination of a grueling six month long season of play stacked on top of decisions made by teams in previous years all for the chance to finally win a championship.

However, not every team has the same chance to win a championship. Some teams are better constructed to win now, some are in purgatory as the middling teams, and some teams are barely capable of winning ten games now as they are planning for the future. Teams like the San Antonio Spurs have a bright future despite their losing record this season and fans are still excited. Contrast this to the Chicago Bulls, a team with limited cap room and no high picks so their fans are understandably less excited for the future of their team.

For our final project, we decided to do a basic sentiment analysis on how the fans of NBA teams feel about their teams currently. A small sect of teams like their chances, others are understandably nervous for their playoff run, and the rest are waiting for the draft to begin later this year. Given how the playoffs just started, there is a heightened amount of discourse within all communities as they reflect on their season. We will use this time period as a good measure to see how teams feel about the season just played and how they feel about the future.

IMPLEMENTATION TOOLS

This project uses the sentiment classification method VaderSentiment, the Reddit API scraping tool Praw, and the clustering method KMeans to separate the positive feelings from the negative.

1 VaderSentiment

VADER is a simple rule-based model for broad sentiment analysis. It combines various qualitative and quantitative methods to result in assessments that tend to outperform human raters and more favorable generalizations across their benchmarks. VADER's effectiveness has been compared to eleven state-of-practice benchmarks: LIWC, ANEW, the General Inquirer, SentiWordNet, and machine learning oriented techniques such as: Naive Bayes, Maximum Entropy, and Support Vector Machines (SVM) and performed just as adequate while being much simpler.

VADER's approach follows an eight step process for constructing its sentiment lexicon. Manually creating a comprehensive sentiment lexicon is very labor intensive. So instead VADER does what many others do: rely on existing lexicons as primary resources that involve basic human heuristics to simplify the process. As an example, an exclamation mark (!) increases the magnitude of the intensity without changing its positive or negative charge. The eight step process (simplified) is as follows:

1) Examine lexical features of existing lexicons
2) Supplement with commonly expressed sentiment (like acronyms or slang)
3) Establish point estimations for each feature candidate
4) Keep gold-standard human-validated lexicons
5) Identify generalizable heuristics
6) Evaluate the impact of grammar and syntax rules on sentiment intensity
7) Establish point estimates by using aggregate data from human raters
8) Compare VADER sentiment to 11 benchmarks-

2 PRAW

PRAW, standing for "Python Reddit API Wrapper", is a python package that allows for easy access to Reddit's API. PRAW can do many things, such as simulating actions of a real reddit account, scraping in post and comment

information from subreddits into text formats, and even moderator actions such as banning users.

3 KMeans

KMeans is a clustering algorithm that aims to place points in k groups which minimizes the sum of squares of points to however many assigned clusters there are. The KMeans problem is typically a version of Lloyd's algorithm (1957) or Elkan's algorithm (2003) which is simply an accelerated version of Lloyd's algorithm.

3.1 Lloyd's Algorithm

The pseudocode for Lloyd's algorithm was provided by Cornell University and is described below:

Input: data $\{x_i\}_{i=1}^{n}$ and k

Initialize C_1, …, C_k

While not converged:

1. Compute $\mu_l = \frac{1}{C_l} \sum_{i \in C_l} x_i f$ for $l = 1, \ldots, k$.
2. Update C_1, …, C_k by assigned each data point x_i to the cluster whose centroid μ_l it is closest to.
3. If the cluster assignments didn't change, we have converged.

Return: cluster assignments C_1, …, C_k

The average complexity of KMeans is O(k n T), where n is the number of samples and T is the number of iterations

The worst case complexity is O(n^(k+2/p)) where n is the number of samples and p is the number of features.

SENTIMENT IMPLEMENTATION

The dataset used for this project is dynamic in the sense that you will mine the data each time it is run rather than import it or store it in a directory. The information being used is post title data from the thirty subreddits focusing on NBA teams. This can be mined from recent posts, top (trending) posts, or the subreddit's all time posts. The user can signify a specific amount of posts to be mined at one time. This limit is typically 500.

1 Obtaining An Authorized Reddit Instance

To do anything with PRAW, you need an instance of the Reddit class. To create an authorized Reddit instance, you need the authentication granted to use the reddit account and then establish an authorized reddit *instance*.

1.1 Authenticating via OAuth

```
import requests
import praw

auth = requests.auth.HTTPBasicAuth('WLcOutFnYEmHQALkCa_8yQ','EddCTpS6sDH8FPGfVnvBZi64fxd-Zw')
data = {
    'grant_type': 'password',
    'username': 'OofInTheChat',
    'password': 'Football5353'
}
```

Figure 1: **OAuth Authentication**

1.2 Making a POST Request for Access Tokens

```
headers = {'User-Agent': 'MyAPI/0.0.1'}
res = requests.post('https://www.reddit.com/api/v1/access_token', auth=auth, data=data,headers=headers)
token = res.json()['access_token']
headers['Authorization'] = f'bearer {token}'
```

Figure 2: **POST Request**

1.3 Initialize Reddit Instance

To create an authorized Reddit instance above the read-only level you need five pieces of information: username, password, client ID, client secret, and a user agent.

```
#initialize reddit API
reddit = praw.Reddit(
    client_id='WLcOutFnYEmHQALkCa_8yQ',
    client_secret='EddCTpS6sDH8FPGfVnvBZi64fxd-Zw',
    user_agent='MyAPI/0.0.1',
    check_for_async=False
)
```

Figure 3: **Initialize Reddit Instance**

2 Using VADER for Sentiment Analysis

2.1 Text Preprocessing

In order to effectively take advantage of VADER, we need to do cleanup of the post data we take in. That involves removing special characters, formatting URLs, and removing extra spaces.

```
#preprocess text
def preprocess_text(text):
    #remove special characters, URLs, and extra spaces
    text = re.sub(r'http\S+', '', text)
    text = re.sub(r'\s+', ' ', text)
    text = re.sub(r'[^\w\s]', '', text)
    return text.strip()
```

Figure 4: **preprocess_text Function**

2.2 Initializing Sentiment Function

Once we have established a method, we need a function that handles VADER's sentiment analysis.

```
#analyze sentiment using VADER
def analyze_sentiment(text):
    analyzer = SentimentIntensityAnalyzer()
    sentiment = analyzer.polarity_scores(text)
    return sentiment['compound']
```

Figure 5: **analyze_sentiment Function**

2.3 Get Sentiments of Desired Posts

Putting the two previous functions together, we create a function responsible for mining the posts and allowing them to be easily accessed by the KMeans Clustering Algorithm.

```
#get sentiment of the latest posts in a subreddit from category
def get_latest_sentiment(subreddit_name, category='new', limit=100):
    try:
        subreddit = reddit.subreddit(subreddit_name)
        if category == 'new':
            posts = subreddit.new(limit=limit)
        elif category == 'hot':
            posts = subreddit.hot(limit=limit)
        elif category == 'top':
            posts = subreddit.top(limit=limit)
        else:
            print("Invalid category. Defaulting to 'new' category.")
            posts = subreddit.new(limit=limit)

        sentiment_scores = []
        post_titles = []
        for post in posts:
            cleaned_title = preprocess_text(post.title)
            sentiment_scores.append(analyze_sentiment(cleaned_title))
            post_titles.append(cleaned_title)

        return post_titles, sentiment_scores
    except BadRequest as e:
        print(f"Bad request error: {e}")
        return None, None
    except Exception as e:
        print(f"An error occurred: {e}")
        return None, None
```

Figure 5: **get_latest_sentiment Function**

3 KMeans Clustering

The KMeans algorithm will group the various sentiment scores into clusters which should allow us to easily be able to pick out which posts fit into which bucket and visualize the amount of posts inside each cluster. This will give us a glimpse at the current sentiment of the 30 NBA teams.

3.1 Clustering

We use the function established above to access the post titles and sentiment scores we found before. We then initialize the clustering algorithm, create a dataframe to organize the data, and then perform calculations of counting and averaging in order to further out analysis.

```
post_titles, sentiment_scores = get_latest_sentiment(subreddit_name, category=post_category, limit=post_limit)

if post_titles and sentiment_scores:
    kmeans = KMeans(n_clusters=3, random_state=0).fit([[score] for score in sentiment_scores]) #kmeans clustering
    sentiment_clusters = kmeans.labels_

    data = {'Post Title': post_titles, 'Sentiment Score': sentiment_scores, 'Cluster': sentiment_clusters}
    df = pd.DataFrame(data)

    cluster_counts = df['Cluster'].value_counts().to_dict() #number of posts in each cluster
    cluster_avg_sentiments = df.groupby('Cluster')['Sentiment Score'].mean().to_dict() #average sentiment score for each cluster

    overall_sentiment = sum(sentiment_scores) / len(sentiment_scores) #overall sentiment score
```

Figure 6: **KMeans Clustering Applied**

3.2 Sentiment Labels

The various sentiment scores are assigned a certain emotion. Very positive scores are given 'happy', very negative scores are given 'angry', and ones in between are given the label 'neutral'.

```
#overall sentiment label
if overall_sentiment > 0.05:
    overall_sentiment_label = 'happy'
elif overall_sentiment < -0.05:
    overall_sentiment_label = 'angry'
else:
    overall_sentiment_label = 'neutral'

cluster_colors = ['blue', 'green', 'red']
```

Figure 7: **Applying the Sentiment Labels**

EXPERIMENTAL RESULTS

Our results were not as accurate as we expected but the experiment did work as intended. Discussion about this will continue in the next section. The following graph represents the r/suns subreddit with the 150 most recent posts. It is classified as angry which was expected since the suns just lost their series by winning 0 out of 4 games and are eliminated from the postseason. The loss occurred a day before this experiment so the recent posts contain the most emotion. The overall sentiment score is -0.02258

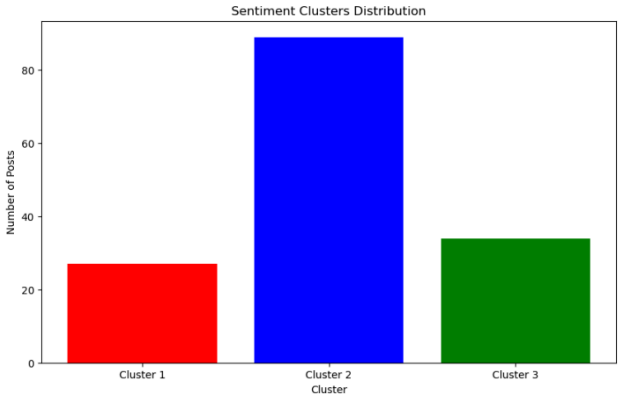

Figure 8: **Number of Posts per Cluster r/suns**

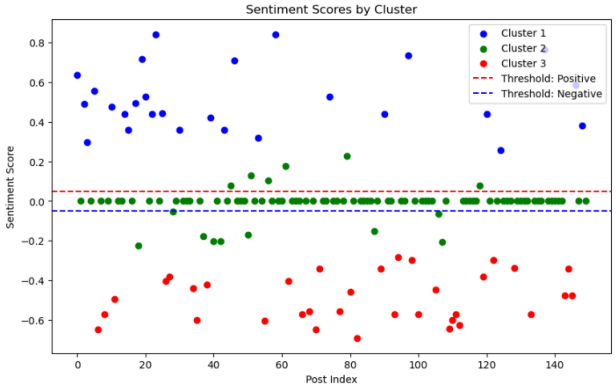

Figure 9: **Sentiment Scores by Cluster r/suns**

The subreddit for the Lakers over the past 450 posts classifies as neutral which is fair considering they started the series off losing 3 straight games but won the last one. The Lakers are also playing the best team in the league and started with two away games so the neutral assessment makes sense. The overall sentiment score is 0.04239

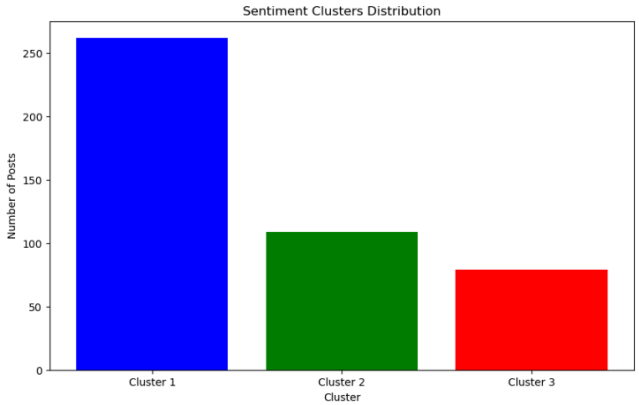

Figure 8: **Number of Posts per Cluster r/lakers**

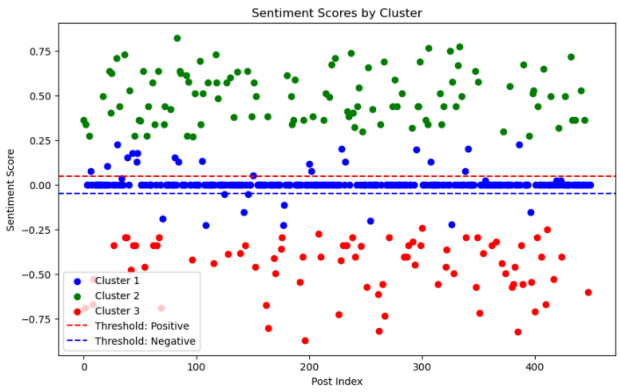

Figure 9: **Sentiment Scores by Cluster r/lakers**

The Timberwolves subreddit classifies as happy with the 150 most recent posts. This makes sense as they just defeated the sun 4-0 and won their first postseason series in 20 years. The overall sentiment score is 0.15184

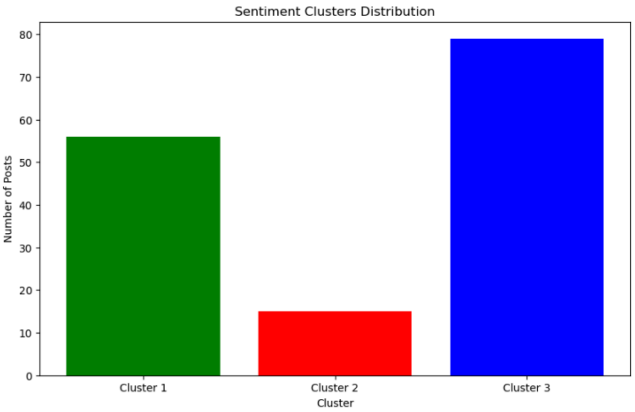

Figure 8: **Number of Posts per Cluster r/timberwolves**

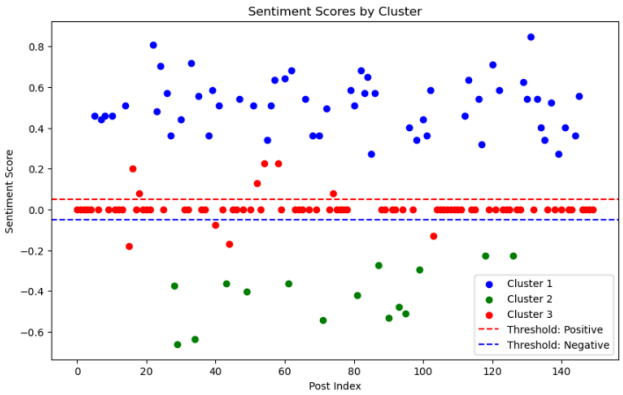

Figure 9: **Sentiment Scores by Cluster r/timberwolves**

ANALYSIS OF RESULTS

The results of these three examples fit our initial expectations but the overall sentiment score wasn't exactly what we were expecting. We expected the suns subreddit to be negative for more than 150 of the most recent posts as

their season just ended but this was not the case. After investigating the subreddit, it seems that the majority of the posts are pictures of players with sarcastic praise used to entice users to banter within the thread. We investigated other subreddits to see if this persisted across them and it did. VADER is used for its simplicity but the small nuances and sarcastic attitude of Reddit users skews how the posts are scored. Furthermore, only the initial posts are captured so most discussion within the subreddit threads are not evaluated which is where most of the emotion is displayed. Curse words used to describe players in a good way are considered negative when in reality they are almost as positive as one can get. Additionally, fans of other teams purposefully post negative things within opposing teams subreddits to entice reaction and spur negative responses, but we didn't capture the responses so this plays little to no role. What shocked us was that nearly all of the subreddits were classified as happy. Our classification for being happy was an overall sentiment score of more than 0.05 since VADER typically ranges from -1 to 1 with -0.05 - 0.05 being neutral. VADER sees users posting 1-2 sarcastic sentences or phrases and a picture of a player within the subreddit as a "positive" sentiment which is why the results are not as expected.

CONCLUSIONS

1 Limitations

A major limitation of this project was PRAW and the Reddit API. PRAW trades its simplistic and easy to use nature for being hamstrung by the limitations of basic Reddit access. For example, you are limited to pulling posts only from the four ways a normal user can view posts. Those being new, top, rising, or hot. This limits you to not being able to sort posts by date, upvote percentage, amount of comments, etc. We want to add onto the project and allow a user to pick a specific year and team to see how the team felt during playoff time in a specific year. We were unable to however due to PRAW not being able to search specific time frames. This is technically possible (using UTC data) but not with the current scope of our project.

There are also limitations with the actual data we are bringing in from Reddit. There is potential for some titles to have (theoretically) no semantic value towards them. Statlines for example showed up a lot of times in the negative category. VADER tried to supplement its lexicon with slang or acronyms but it doesn't catch everything, especially newer trends.

2 Future Improvements

The biggest improvement to be made for this project is both the amount and type of data mined for the sentiment analysis. Currently we are only mining NBA subreddits by hot, new, or top. This could be expanded to other sports such as NFL or MLB which have substantial amounts of posts on their respective subreddits as well. This would require a different and more complex method of mining reddit data and thus PRAW probably wouldn't be used.

There is also potential to add some machine learning models into this as well. While it wouldn't be applicable to real-world problem solving like sentiment analysis and sports marketing it would still be interesting. You could find the overall sentiment of an MLB or NHL team before and around playoff time and use that as a training set. You would have their overall sentiment score and their result for that season. It would be expected that an MLB team who wins the World Series would have a happy sentiment score. You could then use that trained model and use past NBA subreddit posts to 'guess' how far they went into the playoffs. You would want to use MLB or NHL as they both have seven games series similar to the NBA.

3 Significance/Importance

While the significance of THIS project in particular may be of no importance. The overall idea of sentiment analysis in respect to a professional sports team is extremely important. Social media is the main vessel nowadays that fans express their opinions on their team in real-time. While some fans do show up at their team's respective arenas and show discontent through booing (or not even showing up) a vast majority of fans do so online.
Understanding the sentiment of their team, they can learn valuable insights into how their team is currently viewed in the public eye. This can affect how they run engagement, how they tailor their brand, how they treat the athletes playing for them, or potentially alter the costs of tickets and/or merchandise. Ever noticed how a player making a drastic mistake or the team losing a really important game in an embarrassing way can lead to a 'We're Sorry' post online?

This can also work in the positive direction too. If the fanbase is excited the team can lean into that and stroke that fire to increase interaction and attendance to the games. They could also potentially raise prices for games if people are excited enough to forgo the cost to increase revenue. The teams can cultivate strong relationships with their fanbase further reinforcing the good times the team is currently in. The players too, could see all of the good feelings within the fanbase and feel more motivated to perform. Sentiment analysis as a concept is vital to how

teams engage with fans and navigate the sports marketing world.

REFERENCES

[1] Zhang, A., Kapoor, N., & Mehta, P. (2020). Can Generative Language Models Capture Stylistic Patterns of Successful NFL Quarterbacks? A Case Study on Tom Brady and Aaron Rodgers. In Proceedings of the Fourteenth International AAAI Conference on Web and Social Media (ICWSM) (pp. 819-822). Association for the Advancement of Artificial Intelligence (AAAI). https://ojs.aaai.org/index.php/ICWSM/article/view/14550/14399

[2] PRAW Development Team. (n.d.). PRAW: The Python Reddit API Wrapper. GitHub. Retrieved from https://github.com/praw-dev/praw

[3] PRAW Development Team. (n.d.). PRAW documentation. Retrieved from https://praw.readthedocs.io/en/stable/

[4] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[5] R Core Team. (2019). Kmeans function documentation. R Documentation. Retrieved from https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/kmeans

[6] Department of Computer Science, Cornell University. (2022). Lecture Notes: CS 4780 - Machine Learning for Intelligent Systems. Retrieved from https://www.cs.cornell.edu/courses/cs4780/2022sp/notes/LectureNotes04.html#fn4