How People are Talking About the Olympic Sports

Troy Jennings
Carl Ausmees
Corey Vorsanger

COMP 4447: DS Tools 1

August 21, 2021

For a more interactive report experience please go to
https://github.com/cvorsanger/COMP-4447-Final-Project
and launch the binder link

Contents

1 Introduction

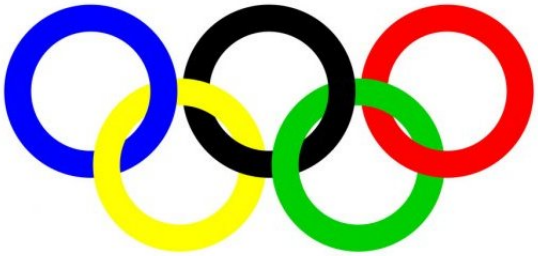
1.1 Research Question

Every four years (five years sometimes) something special happens. No, it is not the world's greatest athletes coming together to showcase elite human athleticism in the Summer Olympics. Rather, it is the millions of people that come together to tweet about these athletes. In a showcase of exceptional human thumbs, these people tell you all you need to know about the Olympics; with 100% accuracy of course.

So what does the twitter-sphere have to say about the Olympics? Are there sports that are being talked about in a better light than others? How does the sentiment of the Olympics change over the course of the event? In this project, we will be investigating tweet sentiment involving Olympic sports to see how they evolve. In the end, the sentiment of specific sports will be analyzed and any trends should be discovered.

1.2 Literature Review

Text sentiment analysis is really nothing new. With the advancements of Natural Language Processing (NLP) techniques building a sentiment analysis tool has become common-



place in a wide variety of applications. While sentiment analysis is not limited to Twitter and can be used for any text data, with its vast amount of text data and the constant addition of data Twitter has been used in many sentiment analysis applications. From using Twitter data to help predict stock data as in Anshul Mittal to the always civil discussions about politics as shown in the Geeks for Geeks article and TSAM. Unsurprisingly, there has been work done in the Olympic domain. The closest work we have discovered documented in a paper was in 2016 Olympic Games on Twitter: Sentiment Analysis of Sports Fans Tweets using Big Data Framework. This paper came out of the Ilamic Azad University in Iran. The writer focuses on Iranian Olympians instead of sports. Using tweets in both English and Farsi they classify tweets as fearful, angry, surprising, sadness, joyful, neutral, and anticipation. We will be only concerned about classifying tweets as negative, positive, or neutral.

While most work follows the same general outline there are differences in sentiment analysis works. For instance, 2016 Olympic Games on Twitter: Sentiment Analysis of Sports Fans Tweets using Big Data Framework uses the WordNet package to handle most of their preprocessing and sentiment analysis model. Geeks for Geeks article uses the TextBlob library for their sentiment analysis. Yalin Yener introduces the Vader sentiment engine in the nltk library.

After review, we have decided to pursue using mostly the Natural Language ToolKit(NLTK) and the Textblob libraries. With these libraries, we should be able to do most of the data cleaning and allows us to use the Vader sentiment engine. More information on Vader will be presented in the Model Creation section.

2 Ingestion

2.1 Motivation

A particular interest of the authors is the ongoing Summer Olympic games. We enjoy watching the top athletes in the world compete in sports that are not on television very often; sports like, gymnastics, swimming, and track. A natural curiosity than was to see how other people view the Olympics.

As social media has grown Twitter has been the go-to platform for the world to express their views. Twitter allows people to express their feelings on just about any subject they desire (for better or for worst). It would make sense then to look at Twitter data to model the sentiment around the Olympics. Luckily, Twitter provides an API for us to query historical tweet data.

2.2 Ingestion

The Twitter API is a relatively easy-to-use API. There are a few restrictions such as rate limits and API types that you do have to consider. For the purposes of this study, the API makes it easy to look at historical tweets containing specific words and hashtags. The following details certain aspects of the Twitter API that pertains to this project. For more information please visit here.

The first step is to get authorization. To get this we must provide the API with our given client key and secret using the POST command with the requests library. Once we get POST we will receive a response from the API we must save.

Listing 1: Authorization

```
import numpy as np
import pandas as pd
import requests
import base64

# Save Authorization Info.
client_key = 'XXXXXXXXXXXXXXXXXXXXXXXXX'
client_secret = 'XXXXXXXXXXXXXXXXXXXXXXXXXXX'
bearer_token = 'XXXXXXXXXXXXXXXXXXXXXXXXXX'
key_secret = '{}:{}'.format(client_key, client_secret) \\
    .encode('ascii')
b64_encoded_key = base64.b64encode(key_secret)
b64_encoded_key = b64_encoded_key.decode('ascii')

# Build API URL
base_url = 'https://api.twitter.com/'
auth_endpoint = base_url + 'oauth2/token'
auth_headers = { 'Authorization': 'Basic {}' \\
    .format(b64_encoded_key),
                'Content-Type': \\
    'application/x-www-form-urlencoded;charset=UTF-8' }
auth_data = { 'grant_type': 'client_credentials' }

# Provide Authorization Info and Save Access Token
response = requests.post(auth_endpoint, headers=auth_headers, \\
```

```
    data=auth_data)
print("Response Status Code: ", response.status_code)
```

As expected we get a response status code of "200". This tells us that we succesfully recieved our response. The next step is to get the *access token* from our authorization response.

A utility function was then created. This function allows us to use the Twitter API to save tweets. The function needs the user's saved access token and allows for the user to specify the number of tweets to pull. The most important parameter of this function is the query parameter. This allows you to filter the tweets you receive by specific hashtags, words, retweets, etc. For this project, we are concerned with tweets containing the hashtags "#olympics" and containing the specific sport.

Listing 2: Tweet Generator

```
def get_tweets(access_token, query, max_tweets=10, tweet_limit=10):

    page_token = None
    tweet_data = []
    search_headers = {
        'Authorization': 'Bearer {}'.format(access_token),
        'User-Agent': 'v2FullArchiveSearchPython',
    }
    # Divides the max tweets into the appropriate number of
    # requests based on the tweet_limit.
    for i in range(max_tweets // tweet_limit - 1):
        search_parameters = {
            'query': query,
            'max_results': tweet_limit,
            'tweet.fields': 'lang,created_at,referenced_tweets,\\
                source,conversation_id'
        }
        # If we reach the 2nd page of results, add a next_token
        #attribute to the search parameters
        if i > 0:
            search_parameters['next_token'] = page_token

        response = requests.get(search_url, headers=search_headers,\\
            params=search_parameters)
        if response.status_code != 200:
            print(f'\tError occurred: Status Code{response. \\
                status_code}: {response.text}')
        else:
            # We need to check for a result count before doing
            # anything futher; if we have result_count we have data
```

```
            if response.json()['meta']['result_count'] > 0:
                tweet_data.extend(response.json()['data'])

                # If a 'next_token' exists, then update the page
                # token to continue pagination through results
                if 'next_token' in response.json()['meta']:
                    page_token = response.json()['meta']['next_token']
            else:
                print(f'\tNo data returned for query!')
                break
        print(f'\t{len(tweet_data)} total tweets gathered')
        time.sleep(1)
    return tweet_data
```

Using the above function we can sample retweets having to do with Olympic gymnastics. Notice that we specify the hashtag "#olympics", the word "gymnastics" and is a retweet in the query parameter. Finally, so we do have to keep requesting data from the Twitter API we save the tweets in a pickle file. This way we can use them later in our analysis (and Twitter will not get mad at us for rate-limits).

Using the "get_tweets" function we tried to pull 5000 tweets for the sports basketball, biking, diving, gymnastics, skateboarding, surfing, track, and volleyball. This was easily done by changing the query parameter to include the name of the sport.

2.3 Sample and Explanation

Alright we have pulled in some tweets. Now lets have a look at a sample of the tweets.

text snippet	id	created_at	conversation_id
cShawn_Shipp_ Thank You, Simone Biles - Headph...	1423371158443941890	2021-08-05 T19:51:43.000Z	1423299151215931392
Unexpected fun fact: Liu Yang originally wa...	1423471713312854017	2021-08-06 T02:31:17.000Z	1423471713312854017
Rhythmic Gymnastics looks insane. #Olympics	1424213988720717828	2021-08-08 T03:40:50.000Z	1424213988720717828
kpkuehler Thank You, Simone Biles - Headphone...	1423796825761456128	2021-08-07 T00:03:10.000Z	1423616706413572103
Simone_Biles you are a wonderful human being ...	1423812938889142273	2021-08-07 T01:07:12.000Z	1423812938889142273

There is a lot of information about tweets that you recieve can recieve from the API. We have narrowed it down to the important ones for this study:

- text - The text of the tweet sent out. This will include user handles, hastags, emojis, and any retweet indicators

- id - The unique id given to the tweet. This allows for twitter to store tweets.
- created_at - When the tweet was tweeted. Given in UTC time.
- conversation_id - The unique id given to twitter conversations. We can use this id to get all tweets in an conversation.

3 Data Cleaning

3.1 Initial Exploration and Cleaning

So, now we have tweets regarding the olympic sports of basketball, biking, diving, gymnastics, skateboarding, surfing, track, and volleyball. All of the tweets were collected into a single dataframe. This will make it easier for us to clean, look at, and use all of the tweets at once. During this process a new variable was created, *sport*. This varibale will track for us the sport that each tweet is talking about and allows us to segment the tweets later when we build our sentiment model.

Using the shape function we find that we have 30554 tweets. Now, time to get our hands dirty and clean the data.

As with all text data, there is a lot of cleaning that can be done. For this project, we will be concentrating on the removal of certain entities such as URL links and hashtags, removal of stopwords, lemmatization, and general NLP text cleaning. After this NLP-specific cleaning, we should be ready to analyze the sentiment of tweets.

Using the below function we were able to use regular expressions to filter the text of our tweets. This function will remove URLs, user handles, hashtags, punctuations, new line characters, and numbers that are in the text of the tweet.

Listing 3: Regular Expression Cleaner

```
1  import re
2  def clean_tweet_text(text):
3      # Define any custom replacement pattern and join into an
4      #aggregated pattern
5      replacements = [
6          r'(@[\w]+)',
7          r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9\\
8  ␣␣␣␣␣␣␣␣␣␣␣␣␣a-fA-F][0-9a-fA-F]))+',
9          r'(#\w+)',
10         r'[\$&+,:;=?@#|\'<>.^*()%!-/]',
11 ␣␣␣␣␣␣␣␣r'\n',
12 ␣␣␣␣␣␣␣␣r'[0-9]+'
13 ␣␣␣␣]
```

```
    aggregate_pattern = r'|'.join(replacements)
    # make regex substitutions in one pass
    clean_text = re.sub(aggregate_pattern, '', text)

    # Return the unicode-stripped text in lowercase
    return clean_text.encode(encoding='ascii', errors='ignore')\\
        .decode('ascii').lower()
```

Stopwords are common words in the English language such as the, an, a, etc. These words are used much in English but convey little meaning. In most NLP applications stop words have very little value and only slow down your model due to their high frequency. While these words shouldn't affect our model that much we will still remove them. The NLTK library has provides a wide assortment of tools to use for NLP. It even includes a list of stopwords we can use. By iterating over our text and comparing the text to the list of stopwords we can remove them for our tweets.

Next, we will lemmatize our tweets. Lemmatization is a process where words are converted to their dictionary forms. For instance, the word "walking" will be analyzed as "walk". Lemmatization will ensure that we have proper words. This is the main reason why we chose to lemmatize instead of stemming as the lexicon in Vader works much better with real words. The NLTK library has the WordNetLemmatizer built-in. We will the WordNetLemmatizer to handle the lemmatization for us.

3.2 Type Conversion

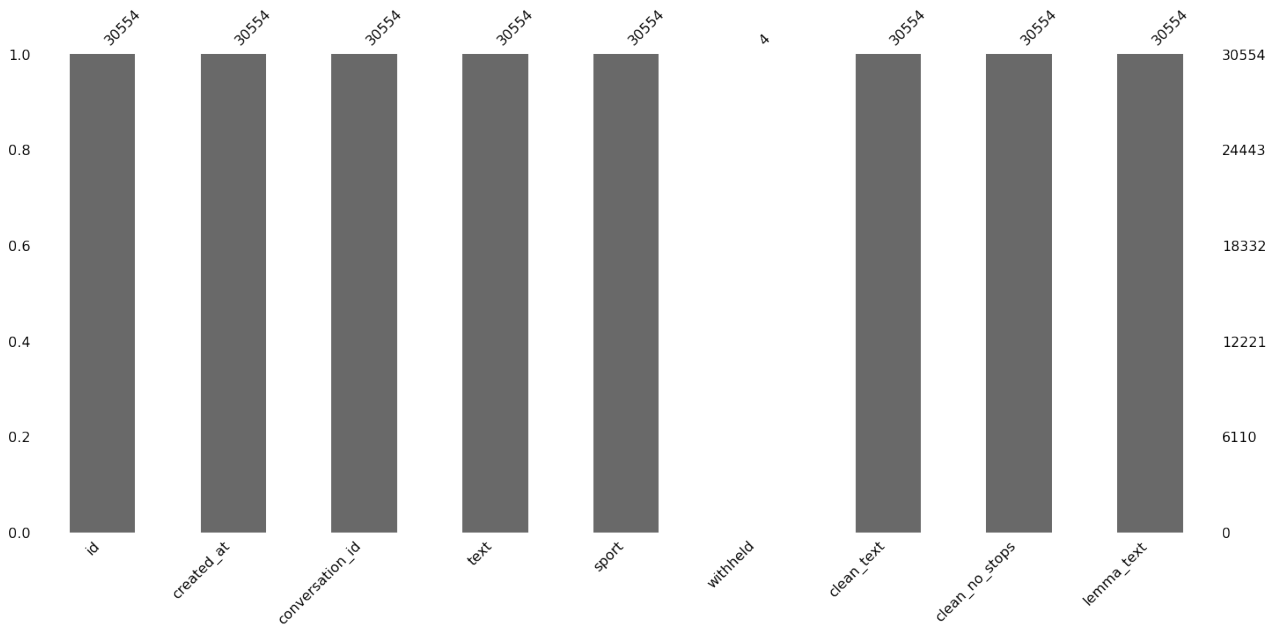
We needed to change a few data types. Thankfully, the tweet text was already given to us as an object (string). To make our analysis easier we changed the *tweet id* and the *conversation ids* of type int from type object. To facilitate the time aspect of our analysis we needed to also covert the *created_at* variable to a datetime object.

This leaves us with these variable types, the correct variable types:

id	int32
created_at	datetime64[ns, UTC]
conversation_id	int32
text	object
sport	object
withheld	object
clean_text	object
clean_no_stops	object
lemma_text	object

3.3 Missing Values

Twitter keeps track of everything about a tweet. This is good news to us because the Twitter API rarely gives you missing values. The below plot shows that there are no important missing values.

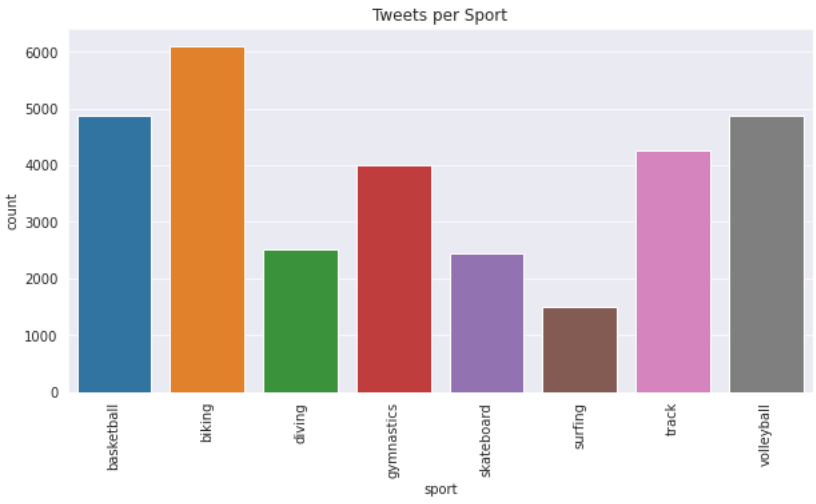


In fact, all of the missing values appear to be in the withheld column. The *withheld* column just tells us in which countries these tweets are not allowed. This is not important in analysis and we will therefore just drop that column.

3.4 Outliers

With text data, it can be difficult to define what an outlier is. Obviously, be can say obnoxiously positive and negative stuff without really knowing anything; this *definitely* doesn't happen on Twitter (wink wink). However, the only real way to detect this is after the sentiment model is built. Also, how do we know what are genuine tweets that should be factored in our analysis and bogus tweets that should not? We do assume that most people that are tweeting about the Olympics are at least watching them. This is also our main target audience in this study; average people watching the Olympics. With it being impossible to detect bogus tweets from the API and our assumption that a vast majority of our tweets originate from our target audience, we decide to analyze all of the tweets we could get. Potentially this could skew our sentiment scores. However, we feel confident that with the number of good tweets this potential skew effect should be drowned out.

3.5 Additional Exploration



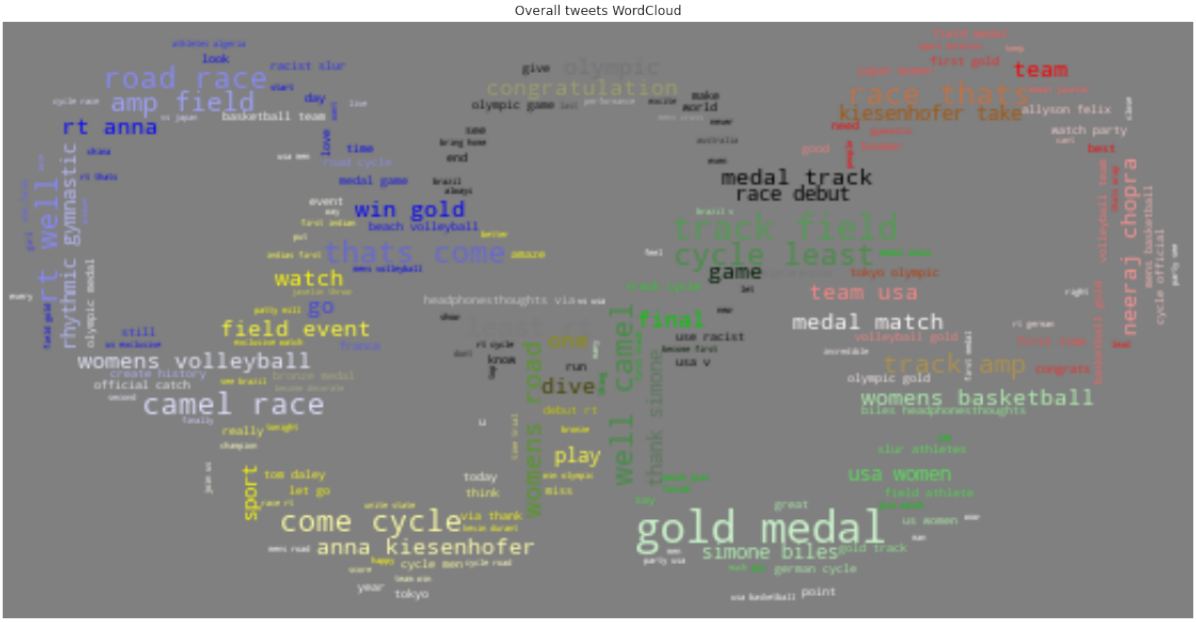
With text data clean, we can do a little more exploration of the data.

As mentioned earlier we tried to get 5000 tweets for each sport. However, this doesn't guarentee that we will recieve 5000 tweets. With the regular twitter API we are limited to tweets created in the past seven days. We did try to upgrade to the acedemic API but were denied, more on the implications of this in **Section 5.2: Future Work**. Using the *sport* column and the viazualization library seaborn, a histogram was made to convey the number of tweets per sport.

We don't have actually 5000 tweets for any sport! the more popular sports sush as basketball, gymnasticsm track, and volleyball do come close. The newer sports such as skateboarding and surfing seem as they are not as talked about. We were surprised by the low number of tweets for diving. While not a super popular sport, diving does seem like it is talked about often when the Summer Olympics come around. Biking does have more tweets then 5000. Initially, we tested our tweets gathering function with a low amount of biking tweets. After we had run our full program we added these initial tweets, which got saved in a pickle file, into the biking tweets pool. Still though, biking appears to be one of the more popular sports.

Wordclouds are another fantastic way to visualize text data. Wordclouds capture the relative frequency of words in a string. The larger the word appears the more frequently it occurs. Words that only appear a couple of times are not pictured in the wordcloud. By concatenating of the tweet text together into one string we can create an overall wordcloud regardless of sport.

The phrase "gold medal" appears to be in a ton of tweets as it looms large in the wordcloud. The name of sports in unsurprisinly said alot, as you can see "basketbal", "track", "volleyball", and "cycling" fairly easily. There are a few athletes that appear in the wordcloud as well. Simone Biles is of course the famous American gymnast. Another name is Anna Kiesenhofer. She is a cyclist that won Austria's first Summer Olympic gold medal since 2004. An interesting word that appears is the word camel. After researching this a little bit, it appears this is in reference to a derogatory term a German cycling coach used in reference to some of the competitors.



Wordclouds can also be made for each sport.

Noticeably skateboarding has very little words. This is due to the small amount of tweets collected. Skateboarding also appears in the surfing wordcloud. A few of these tweets were reviewed and it appears that surfing and skateboarding were talked about in the same tweet. Simone Biles was very popular in the gymnastics tweets. The british athlete Tom Daley appears in the diving wordcloud.

4 Model Creation

4.1 Model

To provide sentiment analysis for tweet data, we utilize two packages: (1) TextBlob and (2) VADER from the NLTK toolkit. TextBlob provides a simple interface for processing text-based data and allows for the calculation of the subjectivity and polarity for a given text, which will aid in sentiment analysis using a set of additional text features. The VADER model is a pre-trained model that uses rule-based values which are especially attuned to sentiments from social media, making it a great choice for overall sentiment analysis.

From the output of TextBlob, we generate two additional features for use in our analysis:

- Polarity: a measure ([-1, 1]) of the sentiment of a text; higher polarity means the text contains more positive sentiment.
- Subjectivity: a measure ([0, 1]) of the opinion and factual information contained in a text; higher subjectivity means the text contains more personal opinion.

From the output of VADER, we generate three additional features for use in our analysis:

- Neg, Neu, and Pos are scores for the ratios for proportions of text that fall into a negative, neutral, and positive sentiment.
- Compound: a score computed by summing the valence scores of each word in the VADER lexicon and normalized to create a composite sentiment score.
- Sentiment: a categorical column that standardizes/generalizes the compound score into positive, neutral, and negative sentiment values.

Listing 4: Authorization

```
# Calculating polarity and subjectivity
olympic_df[['polarity', 'subjectivity']] = olympic_df['lemma_text']\\
    .apply(lambda text: pd.Series(TextBlob(' '.join(text)) \\
    .sentiment))

# Calculating Negative, Positive, Neutral and Compound values
for index, row in olympic_df['clean_no_stops'].iteritems():
    score = SentimentIntensityAnalyzer().polarity_scores(' ' \\
        .join(row))
    neg = score['neg']
    neu = score['neu']
    pos = score['pos']
    comp = score['compound']

    if comp <= -0.05:
        olympic_df.loc[index, 'sentiment'] = 'negative'
```

```
    elif comp >= 0.05:
        olympic_df.loc[index, 'sentiment'] = 'positive'
    else:
        olympic_df.loc[index, 'sentiment'] = 'neutral'
    ## Set the values as columns
    olympic_df.loc[index, 'neg'] = neg
    olympic_df.loc[index, 'neu'] = neu
    olympic_df.loc[index, 'pos'] = pos
    olympic_df.loc[index, 'compound'] = comp
```

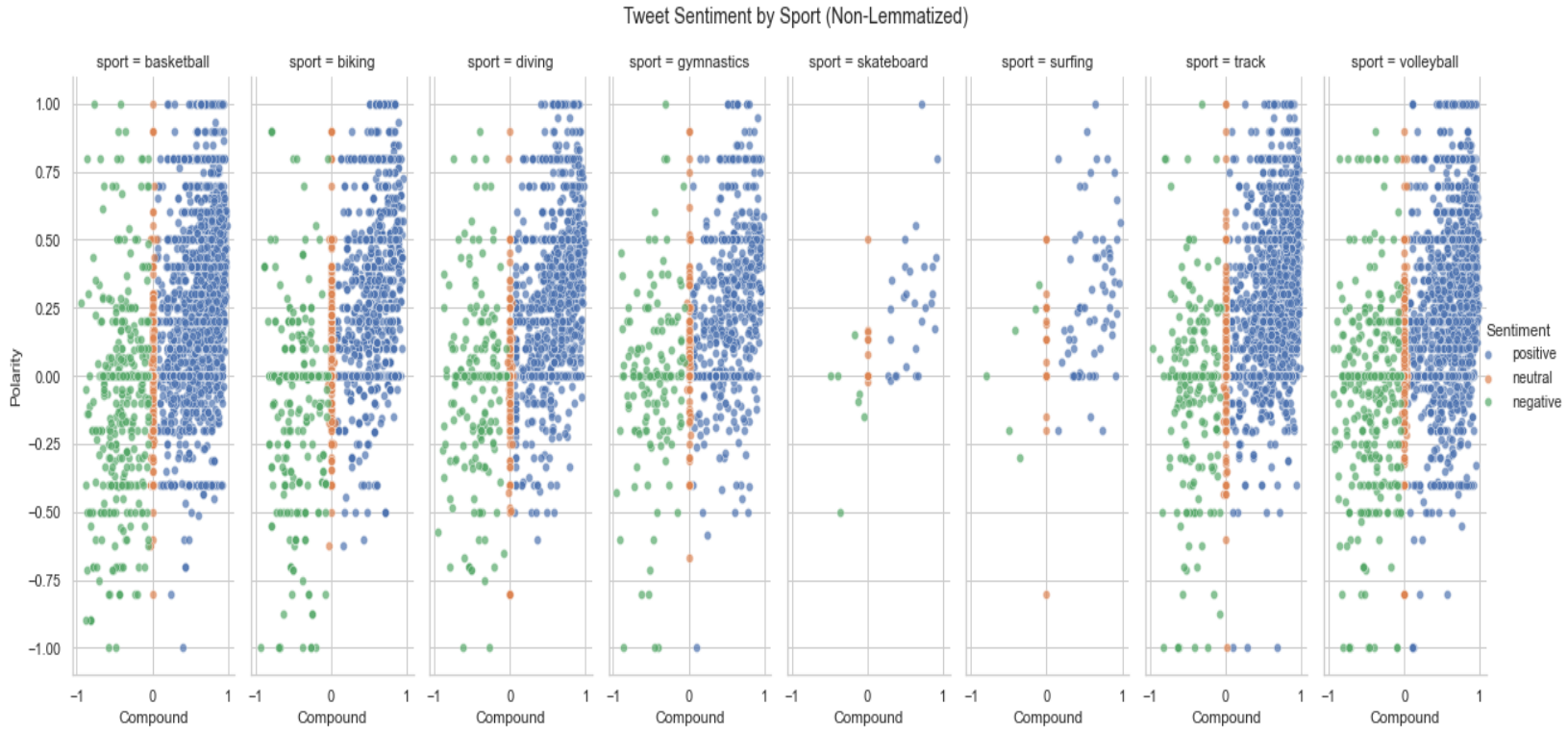
A snippet of our data after running our model will look like:

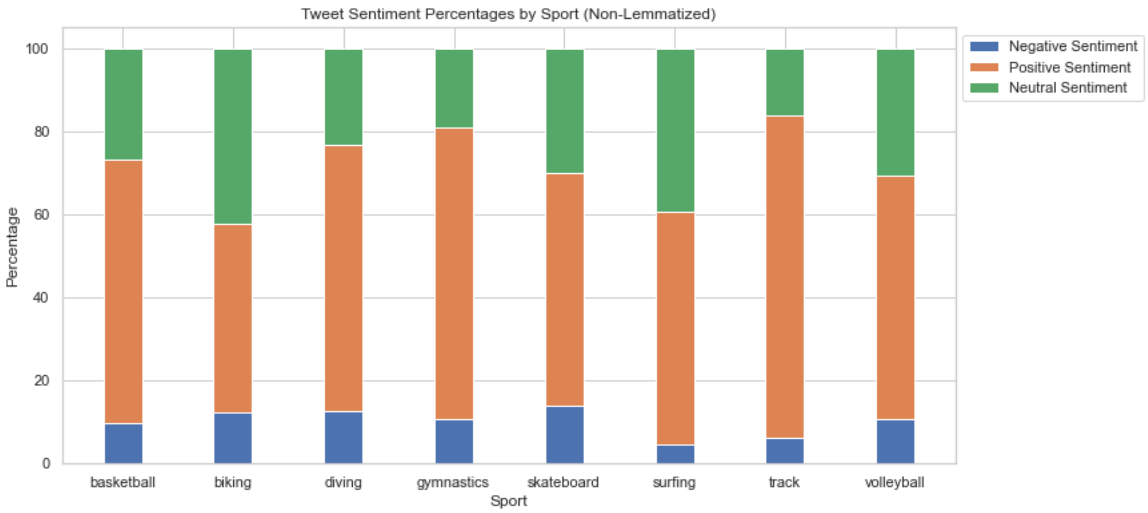
lemma_text	polarity	subjectivity	sentiment	neg	neu	pos	compound
[congratulations, tochelsea, grayon, bring, ho...	1.000	1.000	positive	0.0	0.456	0.544	0.9493
[talkin, noise, podcast, ep, basketball, team,...	0.650	0.650	positive	0.0	0.678	0.322	0.5859
[high, stake, take, lock, still, day, streak, ...	0.144	0.513	positive	0.0	0.865	0.135	0.4215
[thursday, qampaclick, link, bio]	0.000	0.000	neutral	0.0	1.000	0.000	0.0000
[st, three, theme, article, weeks, focus, amaz...	0.500	0.400	positive	0.0	0.598	0.402	0.8555

5 Evaluation and Conclusions

5.1 Results

hoinlongfo


Tweet Sentiment by Sport (Non-Lemmatized)
sport = basketball
sport = biking
sport = diving
sport = gymnastics
sport = skateboard
sport = surfing
sport = track
sport = volleyball
Polarity
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1
0
1
Compound
Sentiment
positive
neutral
negative



5.2 Future Work