Olympic Tweets

August 18, 2021

BY
Troy Jennings
Carl Ausmees
Corey Vorsanger

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 2
1.1 Research Question . . . 2
1.2 Literature Review . . . 2

2 Data Ingestion 3
2.1 Motivation . . . 3
2.2 Ingestion . . . 3
2.3 Sample and Explanation . . . 6

3 Data Cleaning 6
3.1 Initial Exploration and Cleaning . . . 6
3.2 Type Conversions . . . 6
3.3 Outliers . . . 6
3.4 Additional Exploration . . . 6

4 Model Creation 6
4.1 Model . . . 6

5 Evaluation and Conclusions 6
5.1 Results . . . 6
5.2 Future Work . . . 6

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 athletisism 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 then 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 over time. In the end the sentiment of specfic 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 Natuaral Languauge Processing (NLP) techniques building a sentiment analysis tool as 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 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 writter focus on Iranian Olympians instead of sports. Using tweets in both English and Farsi they classify tweets as fearful, angry, surprising, sadness, joyful, neatruel, and anticipation. We will be only concerend about classifying tweets as negative, positive, or neutral.

While most work follows the same general outline there are differences in sentiement 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 decide to prusue 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 Data Ingestion

2.1 Motivation

A particular interest of the authors are 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 curioisty 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 for people to express their feelings on just about any subject they desire (for better or for worst). It would make sence 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

he twitter API is a relative easy to use API. There are a few restictions 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 follow 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.

[ ]:
```
import numpy as np
import pandas as pd
import requests
import base64
# Save Authorization Info.
client_key = 'XXXXXXXXXXXXXXXXXXXXXXXXXX'
client_secret = 'XXXXXXXXXXXXXXXXXXXXXXXXXX'
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 access token.

Lets get the access token from our response and save it in a variable *access_token*.

[3]:
```
json_data =  response.json()
access_token = json_data['access_token']
```

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

[ ]:
```
def get_tweets(access_token, query, max_tweets=10, tweet_limit=10):
    """Retrieve tweets from the recent search API.
        Args
        ----
        access_token (str): A valid bearer token for making Twitter API␣
 ↪requests.
        query (str): A valid Twitter query string for filtering the search␣
 ↪tweets.
        max_tweets (int): The maximum number of tweets to collect in total.
        tweet_limit (int): The number of maximum tweets per API request.
    """
    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 are 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 anaysis (and twitter will not get made at us).

[ ]:
```
# Make the request
q = '#olympics gymnastics -is:retweet'
olympic_tweets = get_tweets(access_token, q, max_tweets=5000, tweet_limit=100)

with open('../data/gymnastics-tweets.pkl', 'wb') as f:
        pickle.dump(olympic_tweets, f)
```

Using the “get_tweets” function we tried to pull 5000 tweets for the sports basketball, biking, diving, gymnastics, skateboarding, surfing, track, volleyball, and cycling. This was easily doen by changing the querey 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 some of the original tweets and the retweets.

```
[ ]: pd.set_option('expand_frame_repr', False)

     original_tweets_df = pd.DataFrame(pd.read_pickle('../data/gymnastics-tweets.
      ↪pkl'))
     original_tweets_df.sample(5)
```

There is a lot of information about tweets that you recieve. For us the most important ones are:

- 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

3.2 Type Conversions

3.3 Outliers

3.4 Additional Exploration

4 Model Creation

4.1 Model

5 Evaluation and Conclusions

5.1 Results

5.2 Future Work