Spatial Sentiment Analysis of NBA Playoffs 2018 Based on Twitter

Team members: Xuefei Liu, Chichiger Shyy, Yuxiao Wang

Abstract:

With the recent sports events such as the Superbowl and the NBA All-Star Game, many people use social media to express their support for the team that they are rooting for. Twitter is one of the social media platforms that many go to, and according to statistics from January 2018, there are over 67 million Twitter users in just the US alone and 330 million users in total. Twitter lets users express their opinions on topics such as politics, sports, games, entertainment, and many more. Users can also connect with others by using hashtags to join in on a conversation or find people with similar interests to follow.

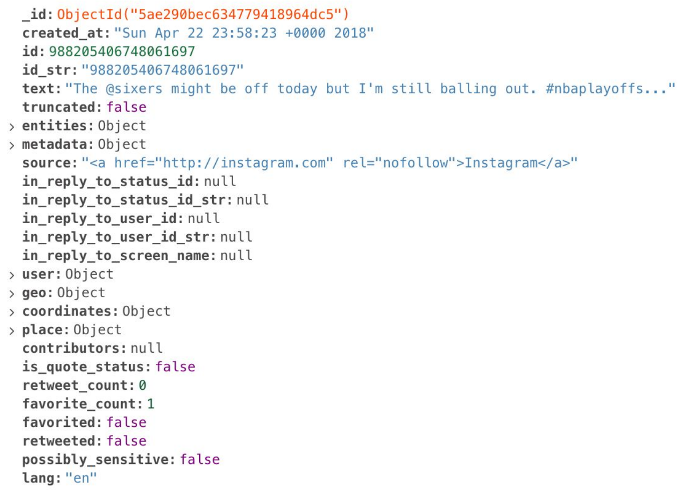
This year, the NBA Playoffs have involved over millions of fans using Twitter to talk about the games and teams they are rooting for. Such a popular and well promoted event such as the annual NBA All-Star Game is worth analyzing about. With the playoffs lasting from the end of April until the first week of June, there will be a large amount of raw data from Twitter to use.

Problem Definition:

The aim of this study is to investigate the association between different states and different sports team affections. We plan to create state level indicator of team affection. For simplicity, only two choices are considered: East and West. Based on the team that they are talking about and which division the team is part of, we will assign each user to East or West. For example, the Celtics, Cavs, and Raptors will be part of the East while the Warriors, Rockets, and Spurs are part of the West. We would like to find the relation or factors in whether local citizens of a state favor their home team or not. Data from social media will help to uncover the team/player preference in certain area, especially Twitter. We would use Twitter as our primary source and restrict to geographic tweets. All the tweets about certain players and teams would be counted toward their divisions. We would use several existing spatio-temporal methods, such as compass, to evaluate the support in different states.

Approach and Method Description:

Due to Twitter Standard API limitation, we could only fetch tweets from the last 7 days, so we did the fetching data method for each day, starting from the last two weeks of April to make sure we had enough data to analyze.



```
  _id: ObjectId("5ae290bec634779418964dc5")
  created_at: "Sun Apr 22 23:58:23 +0000 2018"
  id: 988205406748061697
  id_str: "988205406748061697"
  text: "The @sixers might be off today but I'm still balling out. #nbaplayoffs..."
  truncated: false
> entities: Object
> metadata: Object
  source: "<a href="http://instagram.com" rel="nofollow">Instagram</a>"
  in_reply_to_status_id: null
  in_reply_to_status_id_str: null
  in_reply_to_user_id: null
  in_reply_to_user_id_str: null
  in_reply_to_screen_name: null
> user: Object
> geo: Object
> coordinates: Object
> place: Object
  contributors: null
  is_quote_status: false
  retweet_count: 0
  favorite_count: 1
  favorited: false
  retweeted: false
  possibly_sensitive: false
  lang: "en"
```

As seen in the right picture, Twitter API RESTful provided us with each tweet in the format of JSON data, and the data includes information about the attributes.

We wanted to restrict ourselves to geo-tagged tweets from the USA, which means that we needed to filter and get all the tweets that have the Place ID that is included inside the United States.

We fetched the data with a query that had a specific USA ID (provided by Twitter). This ensured that all our data had a Place data for each tweet so that we could do some analysis based on the geo tagged data in the future.

Collecting Data:

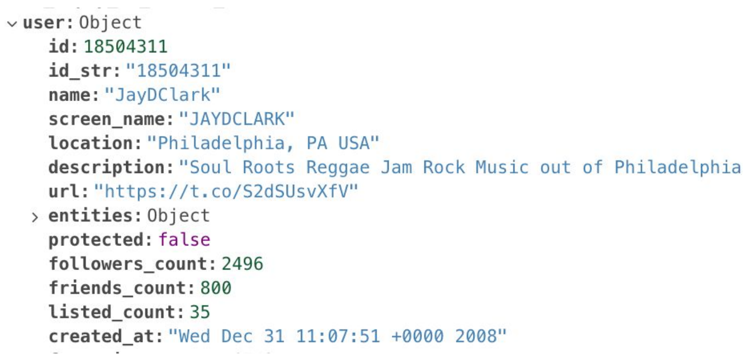
After we created the app on Twitter Developer System and have the keys for the Twitter API, we used Tweepy, an open source library that gave us access to the Twitter API. Tweepy lets us gather tweets from Twitter, and then store it in JSON format.

In 'app.py' file, after we authenticated with Twitter keys, we got each day's tweets using the function 'findTweetsMoreTags', which executed multiple queries with different keywords/hashtags and place id of USA. Then we stored tweets for each day in a MongoDB database. Each Tweet we gathered is in JSON format and has the place information that we will do further combing and filtering data analysis.

Organizing the Data:

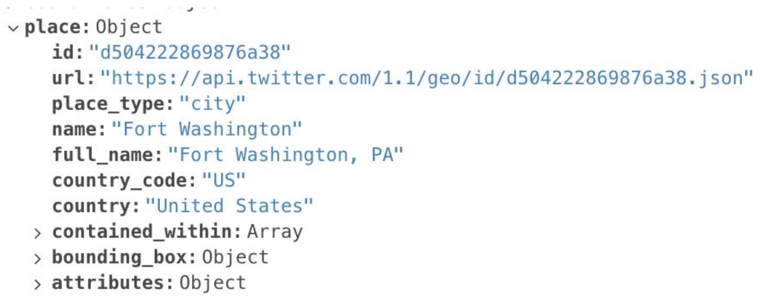
After we have each day's tweets data in different database, we will need to combine these database into one so that we can do split function with the State easily. To do this, inside the 'dataAnalyze.py', we used the function 'dataCombine' to combine all tweets from different days into one database, along with filtering the Tweet unique ID to make sure no Tweet is added multiple times.

Since we want to count how many users in that State rather than how many tweets in that State, we used the user unique id inside the key 'user' in each Tweet object to filter the Tweets that have the same user id. After running the function deleteSameUser", our data is now stored in one collection in Mongo Database 'cs562', which has all tweets that are published by different users with place/geo inside.



```
⌄ user: Object
    id: 18504311
    id_str: "18504311"
    name: "JayDClark"
    screen_name: "JAYDCLARK"
    location: "Philadelphia, PA USA"
    description: "Soul Roots Reggae Jam Rock Music out of Philadelphia
    url: "https://t.co/S2dSUsvXfV"
  › entities: Object
    protected: false
    followers_count: 2496
    friends_count: 800
    listed_count: 35
    created_at: "Wed Dec 31 11:07:51 +0000 2008"
```

The final goal we want is to determine whether the majority of users from each state supports the West or East, and to do that we will need the data split into different States. Then we will use the 'place' key inside each Tweet object to decide which State this user with that Tweet belongs to. In the image to the right, inside 'place' object, the key we need is 'full_name', which is type string. However, not all of the tweet's 'full_name' is in the format 'city, state'. To fix this issue, we made a dictionary 'dictState', where the key is the State full name and the value is the State abbreviation *(ex. 'Alabama':'AL')*. This way we can parse the 'full_name', which has the format 'state full name, USA' *('Alabama, USA')* , via pair the state full name in 'full_name' to the key from 'dictState' into the state abbreviation. Besides this, we also had some places that do not contain any state information, which means we need to use other API to get the city or neighborhood. After some consideration and analysis of the data, we decided to delete the tweets that need further analysis with place because it was only approximately 2% of the data. All these decision are executed in the function 'splitWithState' and finally we had a different collection named 'tweets_state' *(ex. 'tweets_AL')* that have unique users with a tweet in that state.



```
⌄ place: Object
    id: "d504222869876a38"
    url: "https://api.twitter.com/1.1/geo/id/d504222869876a38.json"
    place_type: "city"
    name: "Fort Washington"
    full_name: "Fort Washington, PA"
    country_code: "US"
    country: "United States"
  › contained_within: Array
  › bounding_box: Object
  › attributes: Object
```

Finally, in 'dataMining.py' file, we did a data mining work to parse all state collections in our database to a txt file so that we can use it later for LSI and SVD. To do that, we defined 'getTextFromTweets' function that imports each tweet from each state and analyzes the string to parse it. To avoid that one tweet might have an have empty line inside that will make it become multiple strings, we split it, cleared all unnecessary empty lines, and appended together again for each tweet. In this way, we can also avoid division error when counting empty line in LSI and SVD get zero results.

After all of the analyzing and mining work, we found out WY had no tweets at all and AK had only one tweet. To normalize it, we did not count these two states in our following LSI and SVD processing.

Defining Tweets' Supporting Division:

Our tweets analysis process is done in "tweetsProcess.py". It opens the 48 text files we created in the previous step and uses LSI (Latent semantic indexing) to analyze the tweets. LSI takes documents that are semantically similar, but are not similar in the vector space, and re-represents them in a reduced vector space with higher similarity. It efficiently addresses the problems of synonymy and semantic relatedness.

First of all, we listed out several keyword for supporting west or east, and wrote two queries using these keywords. The hashtags and capital letters do not make a difference since we would tokenize and stem them later.

Division	Keywords
West	Warriors, pelicans, spurs, dub nation, #WarriorsGround, wolves, Rockets, Harden, Curry, Blazers, Anthony Davis, New Orleans, Kevin Durant, KD
East	cavs, pacers, celtics, raptors, wizards, heat, sixers, lebron, boston, philly, #CUsRise, jaylen brown, rozier, Toronto, Wade, Boston, Miami

Table 1: Enriched keywords with division affiliation

Our algorithm first stems all the text files to extract the root of each word and lower all the capital letters in all tweets. This step is done in function stems(x) with nltk python package. It opens a file named x. In x, each tweet is stored in one separate line. We use nltk.TweetTokenizer() to tokenize all tweets. This function is different from the word_tokenize since there are certain grammars people use to post tweets, such as the use of emojis and special symbols. Then, apply nltk.SnowballStemmer to each token to extract the roots. The tweets opened from the text files are stored in a list and then passed to remove(lst, i).

Remove function would remove all the stop words in the list passed from stems. A new text file names 'modx.txt' is created. Once a word in the list does not belong to stop words, it is added to the new file. Once we have iterate through all the tokenized words in one tweet, we start a new line for the next tweet.

The third step is using textmining python package to construct a matrix that count the frequencies of non-stop words in file named x in function documentTermMatrix(x). In this step, we pass the modx files to the parameter. It will return a document-term matrix A1. Based on A1, we continue to construct tf*idf representation called A2 using functions tf(matrix), idf(matrix), and tfidf(tf, idf).

Term frequency (tf) [Hans Peter Luhn 1957] is the frequency of a term that occurs in a document which is proportional to the weight of a term. Inverse document frequency (idf) is a cornerstone of term weighting [Karen Sparck Jones 1972]: The specificity of a term can be quantified as an inverse function of the number of documents in which it occurs. Then Term frequency-Inverse document frequency (tf-idf) is calculated as tfidf(tf, idf) = tf(matrix) * idf(matrix). A high weight in tf*idf is reached by a high term frequency and a low document frequency of the term in the whole collection of documents; the weights hence tend to filter out common terms. As a term appears in more documents, the ratio inside the logarithm approaches 1, bringing the idf and tf*idf closer to 0.

Then we used numpy package (numpy.linalg.svd) to decompose matrix A2 and create the LSI representation. According to theorem [Press 1992]: always possible to decompose matrix A into $A = U \wedge V^T$, where U, ∧, V are unique. U, V are column orthonormal, ∧ is singular valued, non-negative and sorted in decreasing order. This particular decomposition is called singular value decomposition (svd). We then use SVD to compute a new, improved term-document matrix A2' by reducing dimensions (setting less important dimensions to zero).

Then we reduce the dimension k to be equal to 80% of the original number of tweets in one state. Dimension k is rounded. In order to avoid singular matrix error, a state has to have at least two tweets to analyze. Therefore, we leave Alaska (AK) out.

Eventually, we call vectors(f, x) to generate the vector of the given query f. This function will process the queries to generate a vector of the length of the number of words in f with values of 0 and 1, where 0 means the term does not appear in the query and 1 means it does. The cosine similarity between all reduced documents produced by svd and the vector of query is a matrix with n terms, where n is the number of tweets in the file. Each term represent the similarity of the tweet with the query. Therefore, by retrieving the cosine similarity of one file with both 'west.txt' and 'east.txt' (two queries), we are able to define the whether each tweet support east or west.

Once we have finished defining one tweet, we increment the support count by 1. If the tweet stays neutral, both support count is incremented by 1. The final result is inserted into the collection named “stateResult” in the “cs562” database with state name, support side, number of east supports, number of west supports, and number of neutrals.

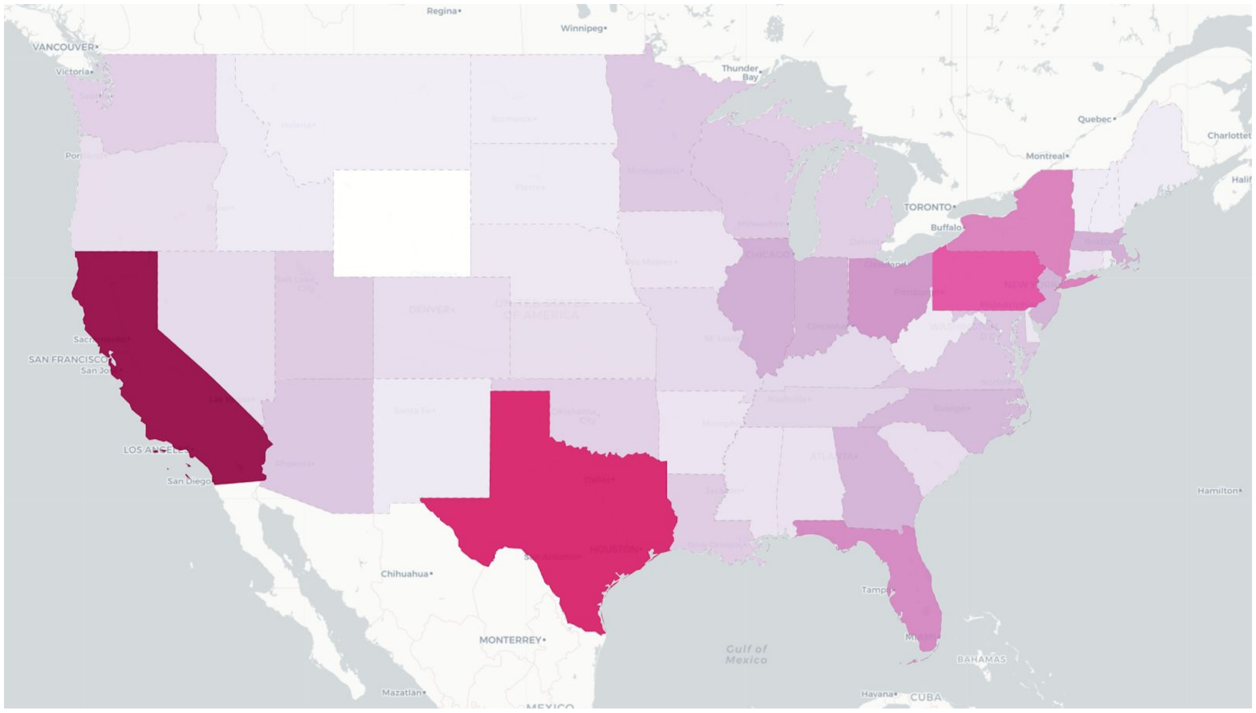
Result:

After we did all of the data analyzing/mining/organizing, and executed LSI and SVD, we had a collection 'stateResult' in our database that contains each state with its support team (east/west), total tweets, east support user number, and west support user number. These are the final results we got from the data from Twitter that we will analyze.

To visualize the two result, inside the 'map.py' file, we used the folium library that import out results in 'stateResult' collection from our database and draw two different maps based on 'usstate.geojson' file (US State).

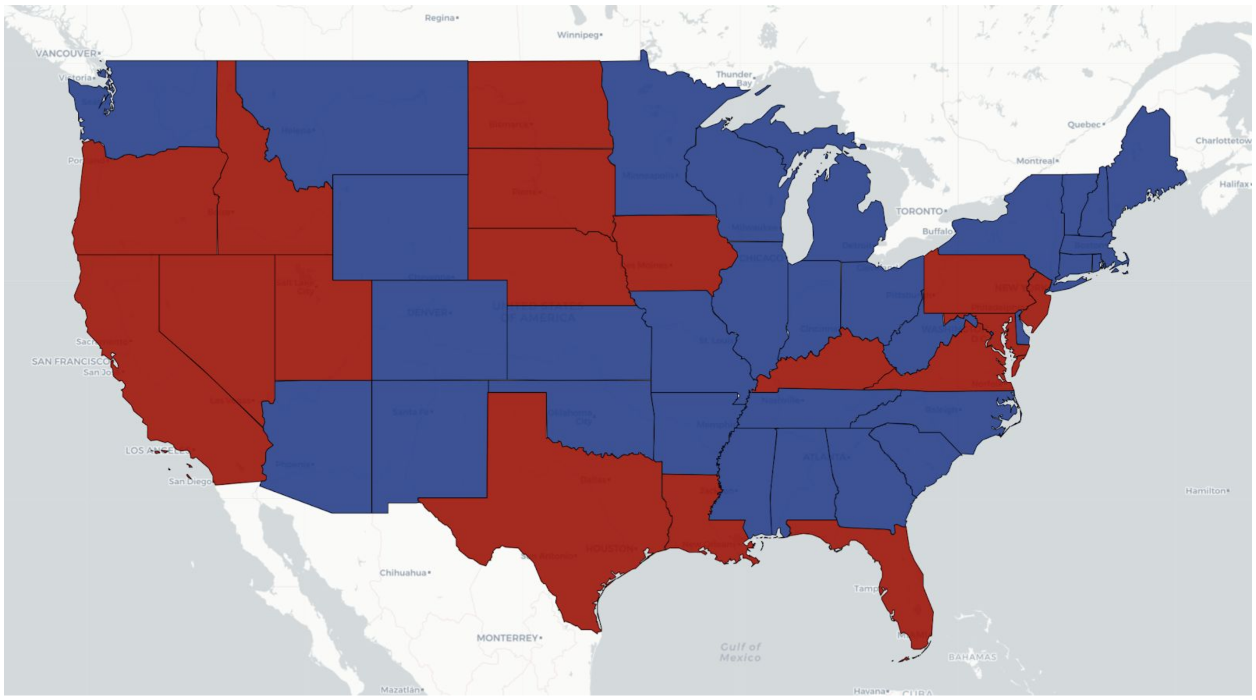
Based on these results, we found out that the top 5 states that have most tweets about NBA playoffs are: CA, TX, PA, NY and FL. (Figure 1)

Figure 1: HeatMap for State Tweets Number about NBA Playoff



After we assigned each state with it's support part (east/west), we found out that there are 30 states who support east and 18 states who support west (Figure 2). From the NBA team state data, there are 10 states have West team and 11 states have East team. Furthermore, from out analyzed result, 4 out of 10 states that have West team are support East, and 3 out of 11 states that have East tram are support West. That is, states that have West team have 60% loyalty to their original side West, and states that have East team have 72.7% loyalty to their original side East.

Figure 2: State Support Map (Red for West, Blue for East)



Conclusion:

From our analysis about Tweets with place metadata, we can easily found out the different result types with geo data. However, Twitter API give us each tweets in JSON format while having different format on the values of some keys. For example, in our report section Organizing the Data, there are lot works on parsing the strings into same format to get the result and filtering later. If possible, Twitter should make some improvement on their data, especially on the place attribution since nowadays the geo/place metadata is the most useful and efficient one in data analyzing.

Due to Twitter API Limitation, our project did not have a huge data set, resulting that in some states only having several results. Beside that, lots of tweets that about NBA Playoffs did not include a location, so we were unable to use it. Because of that, it's normal that our analysis may have some bias on the state support part.

Although we have plenty limitations and hard works on the data collecting and organizing process, we still have a full result based on the Tweets. Some states have less tweets due to the limitation we mentioned above, which should be improved in the future work to give an more accurate result based on larger sample size.

Furthermore, for continuing the data analyzing, we can grab detail NBA state team distribution data and combine with our data to make HeatMap to see the relationship between state that have NBA teams and state that not. Besides this, if we can analyze more data that do not have place metadata, our datasets should have more accurate result.

References:

Deerwester, Scott; Dumais, Susan T.; Furnas, George W.; Landauer, Thomas K.; Harshman, Richard (1990). "Indexing by Latent Semantic Analysis" (PDF). Journal of the American Society for Information Science. 41 (6): 391–407.

GSL Team (2007). "§14.4 Singular Value Decomposition". GNU Scientific Library. Reference Manual.

Hoffman, Thomas (1999), “Probabilistic Latent Semantic Analysis”.Uncertainity in Artificial Intelligence, UAI’99, Stockholm.

Luhn, Hans Peter (1957). "A Statistical Approach to Mechanized Encoding and Searching of Literary Information" (PDF). IBM Journal of research and development. IBM.

Paul, Debjyoti, et al. “Compass: Spatio Temporal Sentiment Analysis of US Election.” Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 17, 2017, doi:10.1145/3097983.3098053.

Spärck Jones, K. (1972). "A Statistical Interpretation of Term Specificity and Its Application in Retrieval". Journal of Documentation. 28: 11–21.