Harrison Tsai, Adam Wang, David Wu

05/06/2014

#hashtag

Introduction

Social media is an ever growing set of data. As people use sites such as Twitter and Facebook, companies are able to gain interesting insights to individuals' habits. Despite the wealth of knowledge available, it is difficult to create insights into people's preferences as they write, tweet, or post about their everyday lives. We believe that extracting data from the raw text of social media can give valuable insights into individual preferences. In this project, we concentrated on data extracted from Twitter because of the character limit and ease to acquire data. For simplicity we chose to work exclusively with a common hashtag used among all the tweets mined, and built a recommendation system on top.

Coachella is a music festival where people from all over the world come to celebrate life and music. It is an annual two-weekend, three-day music and arts festival held at the Colorado Desert in California. The event features a wide range of music, including genres such as rock, indie, hip-hop, and electronic dance music. We chose to concentrate on this event because at the time of the project, #coachella was one of the top trending hashtags since the timing correlated with the project.

This project was inspired by various music services such as Pandora Music, Inc., Spotify Ltd., and StubHub. Both Pandora and Spotify are music stream services that heavily rely on automated music recommendation; however, in both cases, the services rely on direct ratings of likes and dislikes. This binary system limits the data into absolutes, and does not give an indication of how much individuals may like certain music. Additionally, other services, such as StubHub, a secondary ticketing service, do not have direct access to consumers to build recommendations besides through public information or reviews, and is therefore limited in data. Our system bridges the two services by enable a greater breadth for data while enabling the processing of raw data. Additionally, our system does not require active user participation like the music streaming services, and can extract information "on-the-fly" since users will continue using social media to post information about their everyday lives whereas may not be actively using a service to rate products they like.

Tools Utilized

This project was done using ipython notebook, a web-based interactive environment where individuals can easily share code, text, and results with other individuals. Various open sourced python libraries were used to assist in the analysis:

- Pandas
  - http://pandas.pydata.org/
  - "An open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language."
- Spark
  - http://spark.apache.org
  - "fast and general engine for large-scale data processing. "
- Twython
  - http://twython.readthedocs.org/en/latest/
  - "Actively maintained, pure Python wrapper for the Twitter API. Supports both normal and streaming Twitter APIs."
- Scikit-learn
  - http://scikit-learn.org/stable/
  - Machine learning library in python that provides simple and efficient tools for data mining and data analysis. It is built on NumPy, SciPy, and matplotlib
- Data-Driven-Documents (D3.js)
  - http://d3js.org/
  - JavaScript library for manipulating documents based on data.
- Natural Language Toolkit (NLTK)
  - http://www.nltk.org/
  - "A leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning."
- Wordle
  - http://www.wordle.net/
  - A tool to generate word clouds
- ScraperWiki
  - https://scraperwiki.com/
  - Scraper tool that enables quick collection of data and cleaning
- Datamaps
  - http://datamaps.github.io/
  - Customizable SVG map visualizations for the web in a single Javascript file using D3.js

All other analysis was built using the standard python library and shell scripts (primarily bash)

Data Acquisition

We obtained the Twitter dataset using a scraper and pulling from the Twitter API any tweets containing "#coachella" over the course of three weeks (the two weeks of the concert and the week after). Due to rate limits, our dataset was limited to Twitter's preset limits. (available here: https://dev.twitter.com/docs/rate-limiting/1.1)

We outputted the scrapped data into a csv file with this format:

id_str,tweet_url,created_at,text,lang,retweet_count,screen_name,hashtags, query,url,user_mention,media,lat,lng,in_reply_to_screen_name,in_reply_to_status_id

We scraped a total of 241,026 tweets over the course of the 3 weeks. Throughout the project, the tweets and data were stored in pandas dataframes.

In addition to the tweets, we extracted the artists and their Twitter usernames from http://www.coachella.com/lineup/. By using xPath, JavaScript, and JSON, we were able to construct JSONs structured as {name: Twitter handle} and {Twitter handle: name}. This way we can have two datasets available for easy use.

Data Cleaning / Integration

The cleaning process for this project was fairly simplistic. The Twitter data was collected directly into a csv. The integration was simple since Twitter's REST API returns items in a json format. Additionally, Twython had tools that enabled easy data integration and scraperwiki enabled direct downloads into csv formats. For the purposes of this project, we only concentrated on certain attributes of the data collected. We stored the id, screen name, tweet time, raw text, retweet count, hashtags, and geographical data (latitude & longitude) in a dataframe. Additionally, we tokenized each raw tweet and eliminated stop-words. Furthermore, the artists to username dataset required no cleanup since we built the dataset ourselves and formatted it consistently with the Twitter data as we extracted it.

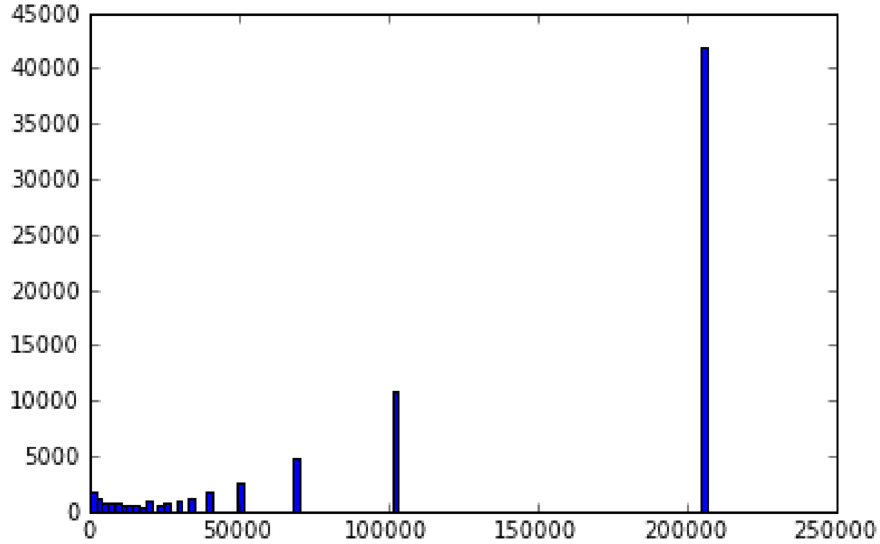
Some issues we encountered during integration involved the clarity of data. The json object the Twitter API returns for a hashtag only includes the first hashtag detected. We had to reconstruct a list of hashtags for each tweet. For this we tokenized each tweet and assumed any word prefixed with ‘#’ was considered a hashtag. Additionally, throughout the project, our analysis was noisy due to poor filtering. To compensate, throughout the project, we updated our stop-words to eliminate irrelevant words from our analysis. Some challenges encountered were non-ASCII characters. For these cases, we manually included the Unicode for these characters in our stop-words set and eliminated them from the set. We had to manually determine the relevance of these characters since some artists would include these characters. One example was the artist by the name “†††”, also known as “Crosses”.

Approaches / Methodology

The steps to build the recommendation system were as follows:

1. Determine the most frequent hashtags and terms
2. Perform sentiment analysis to extract information from raw text
3. Utilize machine learning techniques to use extracted information and build recommendations

First to determine the most frequent hashtags and terms, we constructed a term frequency – inverse document frequency table (TF-IDF). This process was simply tokenizing each tweet and counting the appearance of words. Our tokenization process would also filter out any stop words. Throughout the project, our stop words list was manually updated with our insights from the inverse document frequencies we found. Below is a histogram showing our final distribution of inverse document frequencies:



It is clear that the majority of words used were unique to a specific document. We found that a significant portion of these unique words were tags to other Twitter users. A sample of unique words is:

'rhythmexplorertv', '@mikecmanning', '#grplv', 'lobreen', '@bethneedscoffee',

Additionally, the distribution of inverse document frequency appears somewhat logarithmic such that many words appear only in one tweet, less in two, and even less in three and so forth. This provided some insight to how diverse tweets were while enabling us to see which terms could be considered more informative.

We filtered out common words such as “a”, “she”, “that”, and “you”. This process was manual as we looked at the most common words and determined which words should be considered irrelevant. In general, words that would give no specific insight into sentiments were filtered out (“a”, “she”, “that”) while we kept words that may provide some insight into the tone of the tweet. (“didn’t”, “love”, “okay”)

After filtering out both common and uncommon words, we found 73,492 unique terms across our dataset. The most common terms (in order) were:

‘Coachella’, ‘rt’, ‘justinbieber’, ‘Beyonce’, ‘ChancetheRapper’, ‘#muse’

This provided some interesting insight to our data. Coachella and rt were expected to be very frequent within our terms. ‘Coachella’ is clear since it is the event name. ‘rt’ frequently stands for “retweet” as individuals signal to others to share their tweet. As such, ‘rt’ was also expected since it is a common addition to tweets. ‘justinbieber’ and ‘Beyonce’ were not expected to be most frequent terms since they were not even on the lineup for the music festival; however after some research, we found that Justin Bieber and Beyonce made surprise appearances at the festival. Since these two appeared more frequently than headliners (performers that are star attractions on a program. Typically performs last) such as Chance the Rapper and Muse, we were led to believe people utilized tweets more spontaneously, and to express excitement.

We continued this analysis with hashtags. The top twenty hashtags (excluding ‘coachella’ since every tweet has that from our API call) and their frequencies are displayed below:

Hashtag	Frequency
‘coachella2014’	8511
‘jendall’	4031
‘muse’	3723

‘parallel’	2828
‘heffrondrive’	2707
‘onedrive’	2319
‘help’	2203
‘sorryonedirectionweloveyou’	2192
‘fo’	2175
‘outkast’	1960
‘music’	1595
‘weekend2’	1378
‘illmaticxx’	1207
‘axsfestivals’	1163
‘outkast20’	1106
‘provehitoinaltum’	1095
‘youwouldntunderstand’	1092
‘kevinschmidt’	1011
‘brothers’	1009
‘ad’	962

(A wordmap of these tweets can be found in our Appendix)

This provided some even more interesting insights into the project and led to our decision to perform sentiment analysis. As expected, certain hashtags were expected to be at the top of the frequency list. These hashtags included ‘coachella2014’, headliners such as muse and outkast, and popular promotions such as ‘onedrive’ for Microsoft OneDrive’s promotion at Coachella. The hashtags that truly stood out were: ‘jendall’, ‘heffrondrive’, and ‘sorryonedirectionweloveyou’. These were particularly interesting since these hashtags related to celebrities who were not performers at Coachella and/or did not even attend. Additionally, the lack of the most common terms such as any form of Justin Bieber and Beyonce appearing in hashtag form led us to believe that hashtags could be used to signify a different sentiment than raw text.

Our sentiment analysis utilized the natural language toolkit (nltk) and classified a tweet as positive or negative. We hand classified a set of tweets for our training set, and trained a naïve Bayesian classifier to determine the probability a tweet was ‘positive’ or ‘negative’.

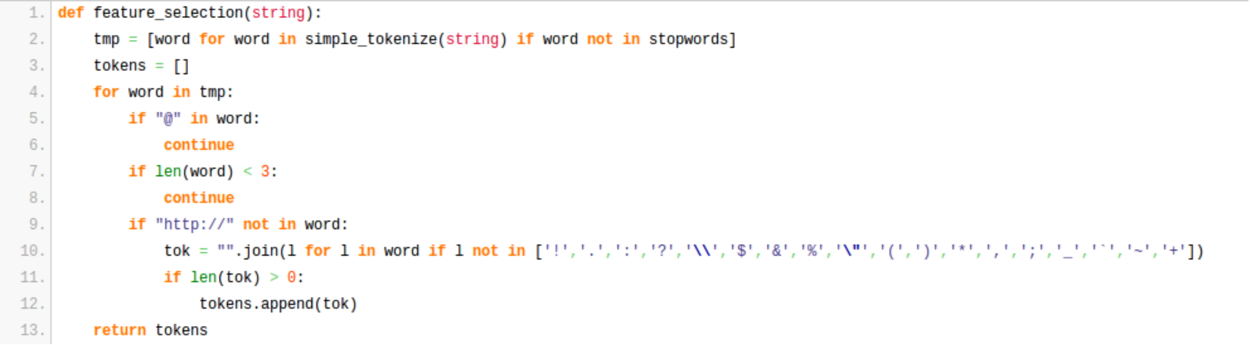
The steps to implement our analysis are as follows:

1. Create training data

This was represented as a list of tuples representing positive or negative tweets: (tweet text, ‘positive’/‘negative’). We hand classified a total of 350 tweets.

2. Identify word features

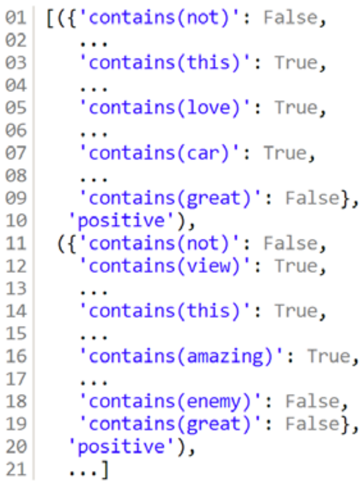
We created a list of every distinct word ordered by the frequency of appearance. This would enable our classifier to decide what features were relevant. To do so, we created a feature extractor, which returned a dictionary indicating what words were in each tweet. Our code is displayed below:



```python
def feature_selection(string):
    tmp = [word for word in simple_tokenize(string) if word not in stopwords]
    tokens = []
    for word in tmp:
        if "@" in word:
            continue
        if len(word) < 3:
            continue
        if "http://" not in word:
            tok = "".join(l for l in word if l not in ['!','.',':','?','\\','$','&','%','\"','(',')','*',',',';','_','`','~','+'])
            if len(tok) > 0:
                tokens.append(tok)
    return tokens
```

We specifically removed tokens that contained @ because they typically mention the artists. These artists are our subjects and including them in sentiment analysis would introduce endogenity and therefore bias our results.

We utilized this function to create a feature dictionary and the sentiment string for each tweet. Our feature extractor would output a list of words, then we would construct a list of tuples which each tuple containing the feature dictionary and the sentiment string for each tweet. One example of our output is displayed below:

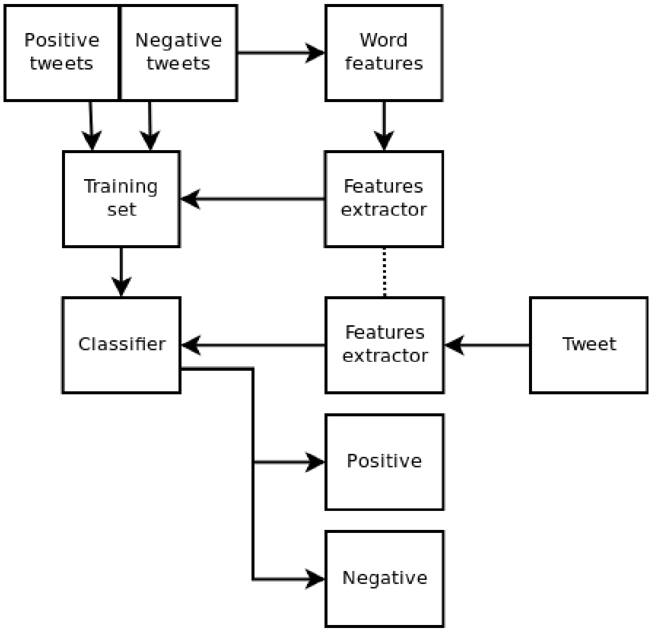


```
[({'contains(not)': False,
   ...
   'contains(this)': True,
   ...
   'contains(love)': True,
   ...
   'contains(car)': True,
   ...
   'contains(great)': False},
  'positive'),
 ({'contains(not)': False,
   'contains(view)': True,
   ...
   'contains(this)': True,
   ...
   'contains(amazing)': True,
   ...
   'contains(enemy)': False,
   'contains(great)': False},
  'positive'),
  ...]
```

3. Train classifier

This simply was trained using the NLTK naïve Bayes classifier

From a high level, our process is represented by the following diagram:

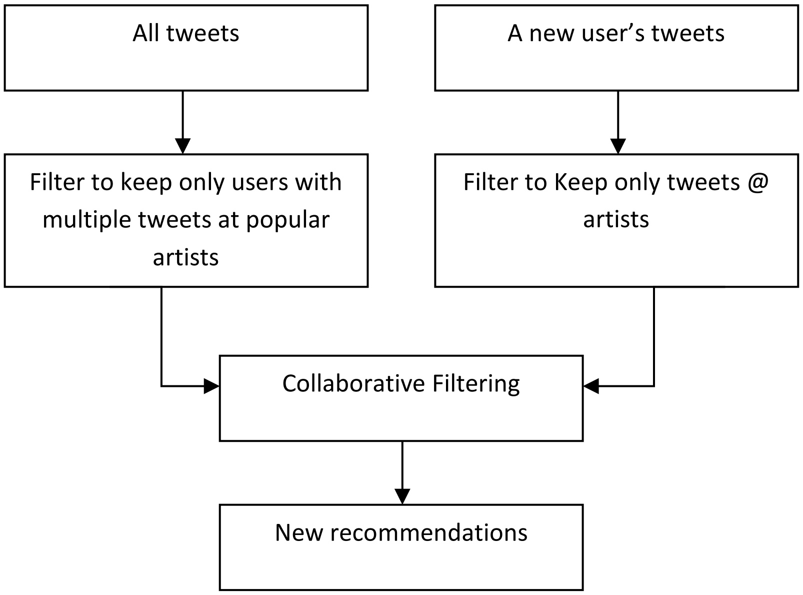


The naïve Bayes classifier utilized the prior probability of each label, which is the frequency of each label in the training set, as well as the contribution from each feature.

Unfortunately our classifier was not very accurate. The vast majority of negative labeled were incorrectly labeled; however, our classifier was able to correctly identify positive labels. Even then, this may have been only because an overwhelming majority of tweets were positive. As a result, for our recommendation system, we assumed positive sentiment indicated a like and disregarded negatively labeled sentiments. This led to interesting insights as individuals either definitively loved certain artists or were indifferent to them. This reasoning is intuitive since people ought to be more likely to tweet positively about the live events they attend than not.

We built additional visualizations which can be found in our appendix at the end of this report.

The data product


All tweets
A new user's tweets
Filter to keep only users with multiple tweets at popular artists
Filter to Keep only tweets @ artists
Collaborative Filtering
New recommendations

Our original approach was to use the probabilities generated from our naïve bayes classifier as ratings and then use collaborative filtering through alternating least squares to create recommendations. As our ratings were not very useful, we had to take another approach. Instead of using probabilities from sentiment, we created a matrix of artists versus screen_names. A cell was '1' if a user mentioned an artist and '0' otherwise. We assigned a score of '1' because we assumed that a user only tweeted an artist if they liked them. We assumed the converse when assigning a 0 (an admittedly weak assumption).

Initially when we loaded this matrix into an ALS model, we had memory errors as there were 14808 users and 194 artists for a total of over

2 million cells. Additionally, this matrix was exceedingly sparse and could have biased our data. To correct for this, we only kept artists that had over 20 tweets at them (79 artists) and only kept users that had greater than 2 tweets at artists (1147). This created around 90000 cells, 87000 of which were 0’s (still pretty sparse).

We loaded this into an ALS model and optimized to find that a rank of 1 was optimal. To see if our model actually worked, we created a fake who only tweeted at a few of electronic artists. Unfortunately, our recommendations turned out just to be the most frequently tweeted at artists. The fact that the top artists got most of the tweets may have been a confounding variable. Additionally, ALS might not work as well for binary ratings.

Conclusion

While analysis on raw text is able to provide some valuable insights, processing raw text is incredibly difficult. We found that our recommendation system may not provide incredibly accurate results, but can provide some qualitative values to any analysis.

Further research needs to be done to properly understand natural language processing. Our model has several underlying assumptions that may bias the data. Additionally, a much larger training set should be used. This process should be automated and possibly crowd-sourced to eliminate some bias, and to quickly grow the dataset. Furthermore, as alluded to earlier, some of the top words or artists found in our TF-IDF analysis seemed arbitrary. This shows the difficulties encountered by our bag-of-words model as it does not necessarily capture the context all the words are in. Some trends/associations may only be made with outside information, beyond the context of just the raw tweet text.

We also need to do further research into what kind of collaborative filtering models are more effective than ALS and if sparsity contributed to our inconclusive recommendations.

This project can be extended into many fields of recommendations. For instance, this project could be extended to produce product recommendations on Amazon by processing people’s posts on social media or restaurant recommendations based on Yelp reviews. These extensions should only be pursued after further research is done to understand natural language processing.

Participation

Harrison Tsai

Scraped Twitter API

Scraped Coachella artist dataset

Hand labeled training set

Built visualizations: Hashtag ratings, sentiment comparisons, word clouds, and maps

Adam Wang

Scraped Twitter API

Worked on sentiment analysis (naïve bayes)

Worked on recommendation system

David Wu

Scraped Twitter API

Cleaned data & filled in missing values with Twython & bash scripts

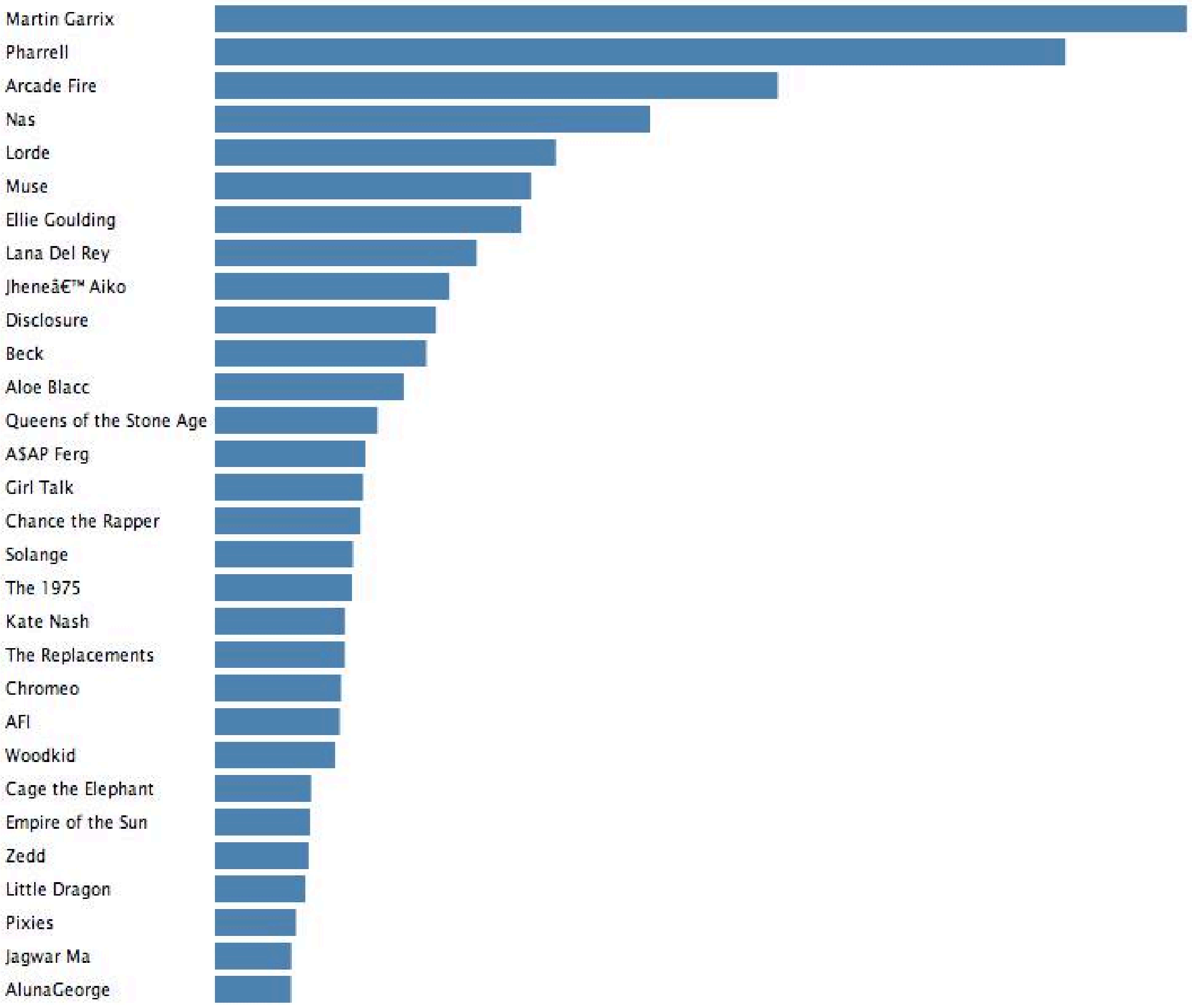
Performed TF-IDF analysis and cleaned data/built stop-words

Performed DBSCAN & visualized location clusters

Worked on sentiment analysis (naïve bayes)

Appendix

Highest Rated Artists 0 – 1

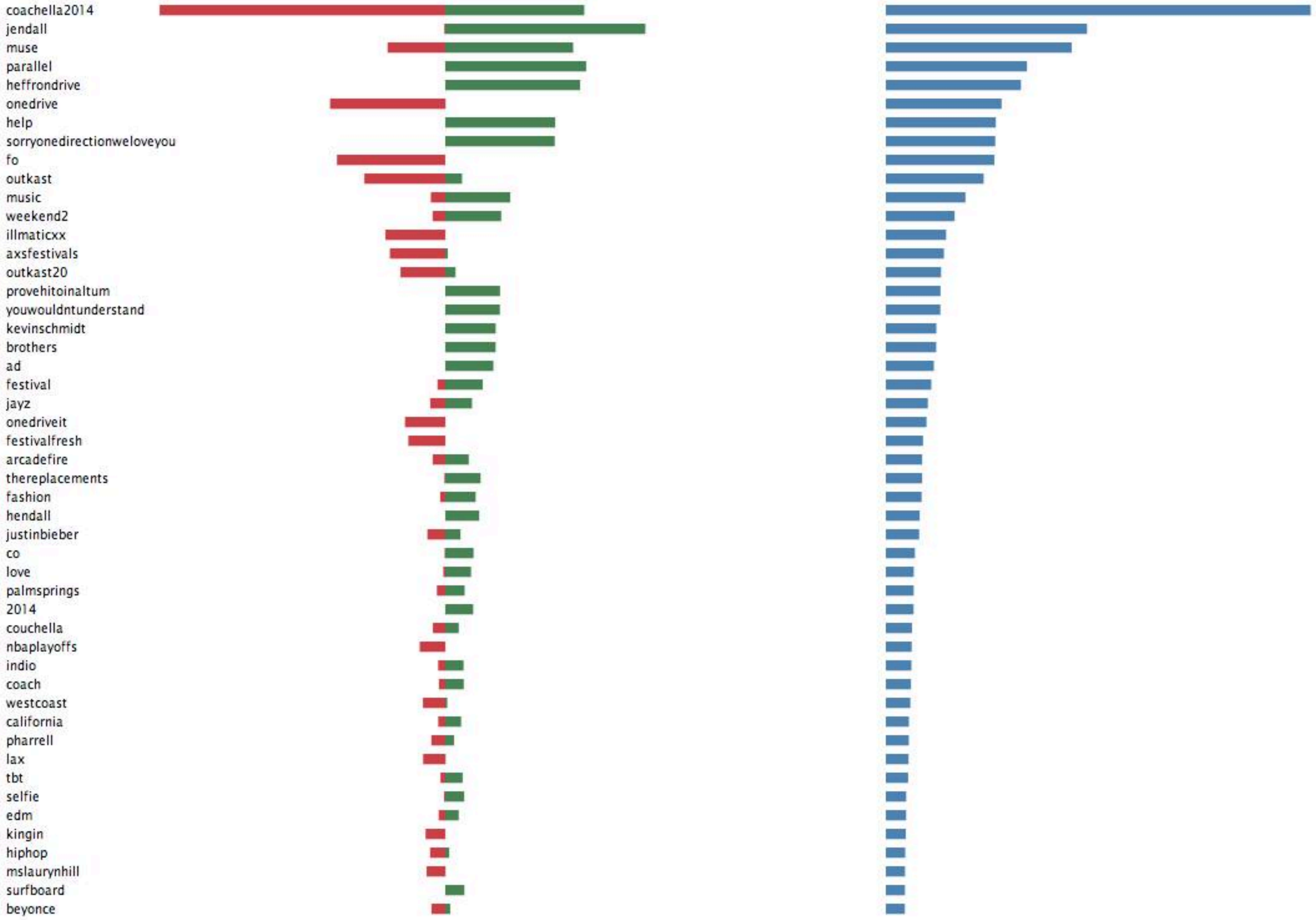


<u>Hashtag Sentiments:</u>

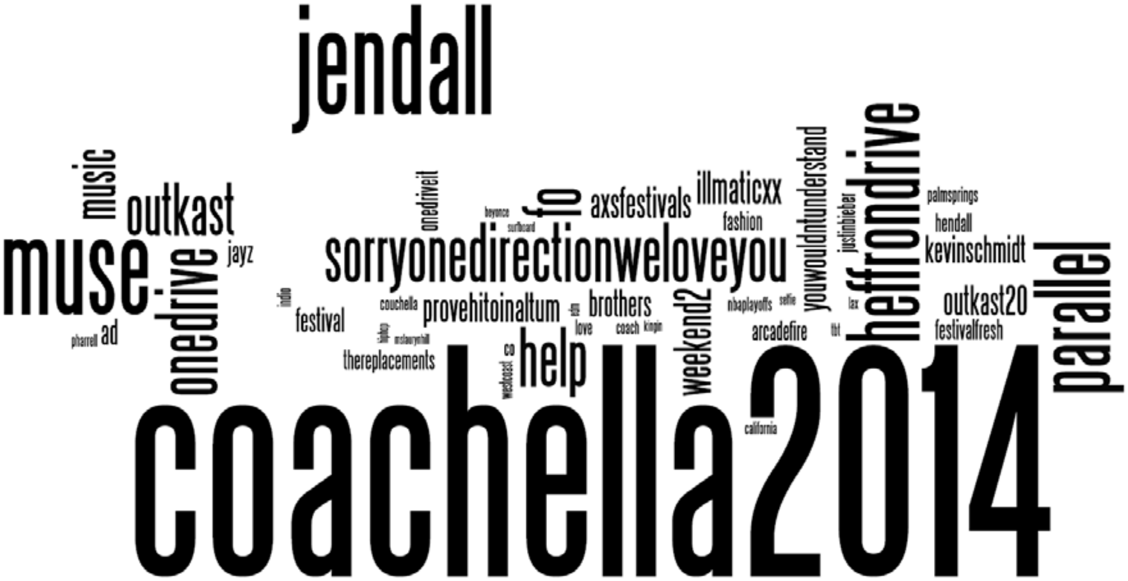
Red = Negative

Green = Positive

Blue = Total

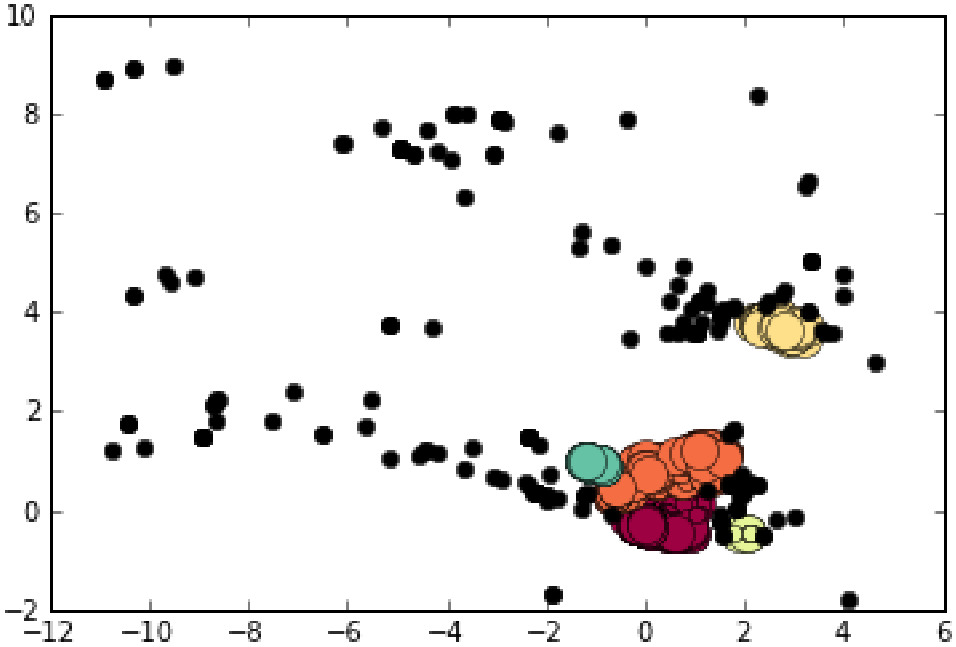


Top hashtags wordmap



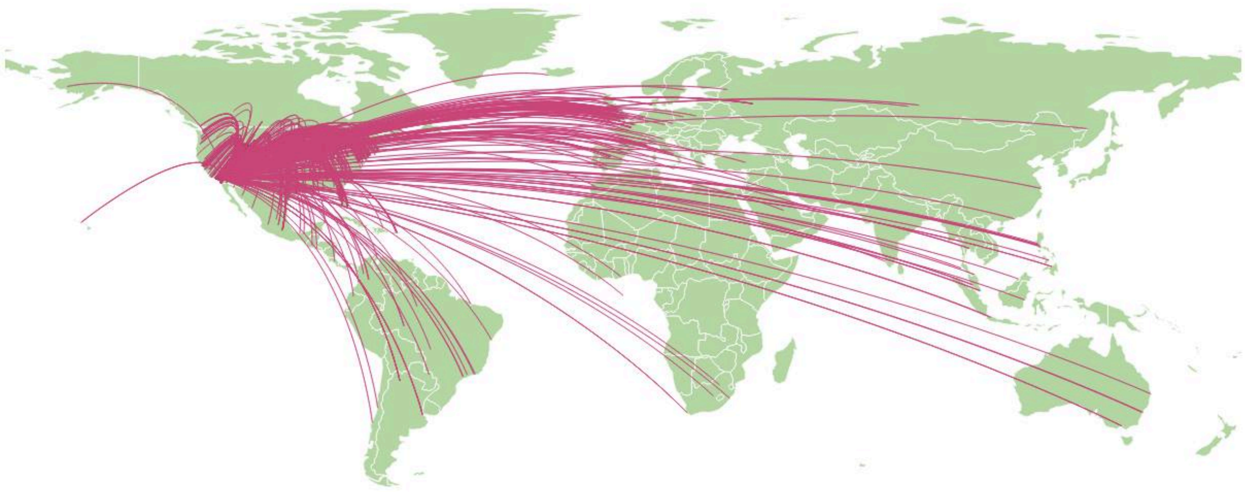
Word map was created to help visualize the most common 50 hashtags found

DBSCAN clustering:



Density-based Spatial Clustering of Application with Noise (DBSCAN) based on geographical data mined. Two major clusters are North America (primarily USA; orange, red, teal, green) and Europe (Western; yellow)

World visualization



USA visualization