Twitter sentiment analysis and identifying opinion of people in different locations

MothiBalaji Srinivasan
3200 Lenox, Atlanta, 30324
929-431-4815
mothicbe@gmail.com

Sai Harika Punyamurthula
1945 Savoy Dr, Atlanta, 30341
385-315-2936
Spunyamurthula1@student.gsu.edu

Abstract

The goal of the project is to collect tweets from different locations and then do sentiment analysis to classify tweet based on their polarity i.e., positive and negative. The positively and negatively classified tweets will then be analyzed and further classify based on Gender. Once, the topics of interest are identified, we can do further analysis to identify correlation in thinking among people of different locations and the correlating how far it has reached the targeted audience.
Keywords : Twitter; classification; machine learning; sentiment; Gender classification.

1 BackGround

Twitter, as a micro blogging website, is one of the biggest web destinations and an open space for people to post and discuss their opinions, thoughts or issues. It has attracted the attention of researchers. Sentiment analysis of tweets is among the hottest topics of research nowadays. State of the art approaches of sentiment analysis present many shortcomings when classifying tweets, when the classification goes beyond the binary or ternary classification[1].

The general procedure for twitter analysis with a machine learning approach is as follows:

- Data extraction we need to extract tweets using appropriate methods for further classification.
- Data cleaning In this step, we first build a corpus and specify source to be character vectors. Then we need to convert whole tweet into lower case letters. Other steps like removing URLs, non-English letters, stopwords (word that do not have lot of meaning like articles, propositions etc.), extra whitespaces are carried out.
- Sentiment Analysis is done to classify each of the tweets based on polarity as positive, negative or neutral using a bag-of-words approach[2]. For sentiment analysis from text, usually, the first step is feature extraction which includes generating feature vectors for classification purpose [3].

2 PROBLEM DESCRIPTION

The data will be extracted based on the location from twitter. Project aims to understand the trends or major topics of discussion among the people from different locations. Few of the topics might include public health issues, trending products, technology related topics. Our aim is to achieve the following:
i)Impact is always Important for any Action.
ii)The goal of the project is to collect tweets from different locations and then do sentiment analysis to classify tweet based on their polarity i.e., positive and negative and neutral.
iii)After polarity classification we aim to understand gender demographics related to a particular topic and location.
iv)We intend to perform this analysis on data from multiple locations and then compare them and identify any correlation in the public opinion among different locations.
v)Capturing the INTEREST of a User based on personal opinion with LIVE tweet Data.

3 POTENTIAL APPROACH

3.1 Workflow

We propose a method which involves the following modules,
a) After Identifying the location, data is extracted.
b) From the extracted data, we need to do classification based on polarity i.e., each tweet can be assigned the label as positive, negative or neutral.
c) We are finding the gender of the people using their names
d) Classifying the data based on the gender inclination towards the topic.

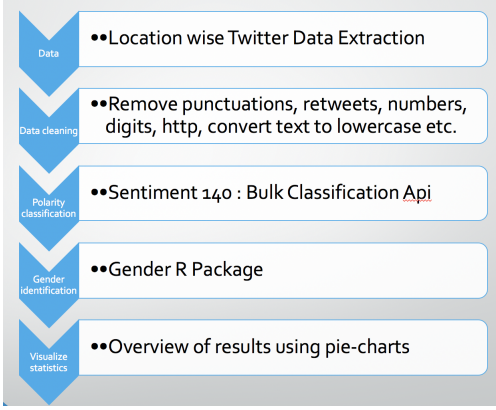

Figure 1: WorkFlow of Our Approach

3.2 Data Extraction module

In our method use Twitter API to extract data, It is very reliable, and available system for efficiently collecting, aggregating and moving large amounts of data from many different sources to a centralized data store and this is achieved with the help of R.

3.2.1 Twitter Service

A. Twitter Login
1)Present users with easy to find options to log into and out of Twitter, for example, via the OAuth protocol or Twitter Kit.
2)Provide users without a Twitter account the opportunity to create a new Twitter account.
3)Display the Connect with Twitter option at least as prominently as the most prominent of any other third party social networking sign-up or sign-in marks and branding appearing on your Service.

B. Social Updates
1)If you allow users to create social updates from your own social service or a third party social networking, micro-blogging, or status update provider integrated into your Service (”Update”), you must display a prominent option to publish that content to Twitter.
2)If Updates are longer than 140 characters or not text, you must display a prominent link to publish that content to Twitter and: URLs must direct users to the page where that content is displayed. You may require users to sign in to access that page, but the content must not otherwise be restricted from being viewed.
3)URLs must not direct users to interstitial or intermediate pages.

C. Twitter Identity
1)Once a user has authenticated via Connect with Twitter via your Service, you must clearly display the users Twitter identity via your Service. Twitter identity includes visible display of the users avatar, Twitter user name and the Twitter bird mark.
2)Displays of the users followers on your Service must clearly show that the relationship is associated with the Twitter Service.

3.3 Extraction logic:

The Streaming APIs give developers low latency access to Twitters global stream of Tweet data. A streaming client will be pushed messages indicating Tweets and other events have occurred, without any of the overhead associated with polling a REST endpoint.

If your intention is to conduct singular searches, read user profile information, or post Tweets, consider using the REST APIs instead.
Twitter offers several basic streaming endpoints, each customized to certain use cases.

An app which connects to the Streaming APIs will not be able to establish a connection in response to a user request, as shown in the above example. Instead, the code for maintaining the Streaming connection is typically run in a process separate from the process which handles HTTP requests.

The streaming process gets the input Tweets and performs any parsing, filtering, and/or aggregation needed before storing the result to a data store. The HTTP handling process queries the data store for results in response to user requests. While this model is more complex than the first example, the benefits from having a realtime stream of Tweet data make the integration worthwhile for many types of apps.

3.4 Data Collection

Connecting
To connect to the Streaming API, form a HTTP request and consume the resulting stream for as long as is practical. Our servers will hold the connection open indefinitely, barring server-side error, excessive client-side lag, network hiccups, routine server maintenance or duplicate logins. The method to form an HTTP request and parse the response will be different for every language or framework, so

consult the documentation for the HTTP library you are using.
Some HTTP client libraries only return the response body after the connection has been closed by the server. These clients will not work for accessing the Streaming API. You must use an HTTP client that will return response data incrementally. Most robust HTTP client libraries will provide this functionality. TheApache HttpClientwill handle this use case, for example.

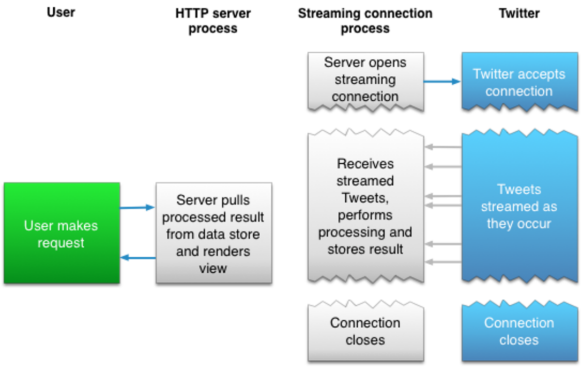

Figure 2: Differences between Streaming and REST

Streaming API request parameters
1) Delimited
2) StallWarnings
3) FilterLevel
4) Language
5) Follow
6) Track
7) Locations
8) Count
9) With
10) Replies
11) StringifyFriendId
Use the following request parameters to define what data is returned by the Streaming API endpoints

Language This parameter may be used on all streaming endpoints, unless explicitly noted.
Setting this parameter to a comma-separated list of BCP 47 language identifiers corresponding to any of the languages listed on Twitters advanced search page will only return Tweets that have been detected as being written in the specified languages. For example, connecting with language=en will only stream Tweets detected to be in the English language.

Follow A comma-separated list of user IDs, indicating the users whose Tweets should be delivered on the stream. Following protected users is not supported. For each user specified, the stream will contain:

Tweets created by the user. Tweets which are retweeted by the user. Replies to any Tweet created by the user. Retweets of any Tweet created by the user. Manual replies, created without pressing a reply button (e.g. @twitterapi I agree). The stream will not contain:

Tweets mentioning the user (e.g. Hello @twitterapi!). Manual Retweets created without pressing a Retweet button (e.g. RT @twitterapi The API is great). Tweets by protected users.

Track A comma-separated list of phrases which will be used to determine what Tweets will be delivered on the stream. A phrase may be one or more terms separated by spaces, and a phrase will match if all of the terms in the phrase are present in the Tweet, regardless of order and ignoring case. By this model, you can think of commas as logical ORs, while spaces are equivalent to logical AND.
The text of the Tweet and some entity fields are considered for matches. Specifically, the text attribute of the Tweet, expandedUrl and displayUrl for links and media, text for hashtags, and screenName for user mentions are checked for matches[5].

Each phrase must be between 1 and 60 bytes, inclusive.

Exact matching of phrases (equivalent to quoted phrases in most search engines) is not supported.

Punctuation and special characters will be considered part of the term they are adjacent to. In this sense, hello. is a different track term than hello. However, matches will ignore punctuation present in the Tweet. So hello will match both hello world and my brother says hello.

UTF-8 characters will match exactly, even in cases where an equivalent ASCII character exists. For example, touch will not match a Tweet containing touche.

Non-space separated languages, such as CJK are currently unsupported.

URLs are considered words for the purposes of matches which means that the entire domain and path must be included in the track query for a Tweet containing an URL to match. Note that displayUrl does not contain a protocol, so this is not required to perform a match.

Twitter currently canonicalizes the domain www.example.com to example.com before the match

is performed, so omit the www from URL track terms.

Finally, to address a common use case where you may want to track all mentions of a particular domain name (i.e., regardless of subdomain or path), you should use example com as the track parameter for example.com (notice the lack of period between example and com in the track parameter). This will be over-inclusive, so make sure to do additional pattern-matching in your code. See the table below for more examples related to this issue.

Locations A comma-separated list of longitude,latitude pairs specifying a set of bounding boxes to filter Tweets by. Only geolocated Tweets falling within the requested bounding boxes will be includedunlike the Search API, the users location field is not used to filter Tweets[8]. Each bounding box should be specified as a pair of longitude and latitude pairs, with the southwest corner of the bounding box coming first. For example: Bounding boxes do not act as filters for other filter parameters. For exampletrack=twitterLocations=-122.75,36.8,-121.75,37.8would match any Tweets containing the term Twitter (even non-geo Tweets) OR coming from the San Francisco area. The streaming API uses the following heuristic to determine whether a given Tweet falls within a bounding box: If the coordinates field is populated, the values there will be tested against the bounding box. Note that this field uses geoJSON order (longitude, latitude). If coordinates is empty butplaceis populated, the region defined inplaceis checked for intersection against the locations bounding box. Any overlap will match. If none of the rules listed above match, the Tweet does not match the location query. Note that thegeofield is deprecated, and ignored by the streaming API. If you would like to excludeplacematches or only include places which fall completely within the bounding box, your code will have to perform an additional filtering step after reading the filtered stream. Note that native Retweets are not matched by this parameter. While the original Tweet may have a location, the Retweet will not.

Parameter value	Tracks Tweets from...
-122.75,36.8,-121.75,37.8	San Francisco
-74,40,-73,41	New York City
-122.75,36.8,-121.75,37.8,-74,40,-73,41	**San Francisco** OR New York City

Figure 3: Extraction logic for Twitter API

4 Classification

Sentiment of Twitter messages are classified as either positive or negative with respect to a query term. We will clear those tweets with neutral values.
We will use the Sentiment 140 which is very good resource from Stanford University. It helps to classify the tweets based on polarity.
The following response denotes the polarity,
0 Negative
2- Neutral
4- Positive.
Based on the response values which are received from JSON we can classify our tweet data[6].
You can include a query parameter, which specifies what the tweet is about. Its recommended that you provide the query term. The query parameter helps us prevent certain keywords from influencing sentiment. For example, if the query was Horrible Bosses (referring to the 2011 movie), supplying the query parameter would help us know that horrible is part of the query and shouldnt contribute to the calculated sentiment.
The words that people use to express sentiment can vary greatly between topics. For example, consider the word scary. Its positive if you find Silence of the Lambs is scary, but negative if your Toyotas brakes are scary. By default, we use a generic sentiment model that works okay across different domains. But, if you supply a topic, we use a domain-specific classifier on the backend that can provide better accuracy.

Bulk Classification Service (JSON) - Recommended
This is the recommended way to utilize our API. You can send thousands of tweets per request, and receive the responses in bulk.

Request Parameters:

- text: The text you want to classify. This should be URL encoded.

- query: The subject. This should be URL encoded. (optional)
- callback: The callback function (optional). Leave this out if you want a pure JSON object returned.
- language: The language of the text. Valid values are:
- en (English: default) es (Spanish) auto (autodetect the language).

Response Data: text: original text submitted query: original query submitted polarity. The polarity values are: 0: negative 2: neutral 4: positive

5 Gender Identification

Base Idea: There are online api which helps to give us the Gender based on the given name,

Figure 4: Extraction logic for Twitter API

This was the driving step to integrate these kind of API for our application work. We used Genderize.io in our project which is a product from MIT license.

Determine the gender of a First Name Genderize.io determines the gender of a first name. Use the API for analytics, ad targeting, user segmenting etc. It utilizes big datasets of information, from user profiles across major social networks and exposes this data through its API. The response includes a certainty factor as well.

To achieve more qualified guesses, it is also possible to use localization filters to retrieve a guess based only on data for a certain country or language. It's recommended to always use a filter if you have the needed data, since naming can rely heavily on demographics.

At the moment, the database contains 216286 distinct names across 79 countries and 89 languages.

Limitation The API is free, but limited at 1000 names/day.

ROpenScience At rOpenSci we are creating packages that allow access to data repositories through the R statistical programming environment that is already a familiar part of the workflow of many scientists. Our tools not only facilitate drawing data into an environment where it can readily be manipulated, but also one in which those analyses and methods can be easily shared, replicated, and extended by other researchers.

Description

This function predicts the gender of a first name given a year or range of years in which the person was born. The prediction can use one of several data sets suitable for different time periods or geographical regions. See the package vignette for suggestions on using this function with multiple names and for a discussion of which data set is most suitable for your research question. When using certains methods, the genderdata data package is required; you will be prompted to install it if it is not already available [7].

Usage

gender(names, years = c(1932, 2012), method = c("ssa", "ipums", "napp", "kantrowitz", "genderize", "demo"), countries = c("United States", "Canada", "United Kingdom", "Germany", "Iceland", "Norway", "Sweden"))

Arguments

names - First names as a character vector. Names are case insensitive.

years - The birth year of the name whose gender is to be predicted. This argument can be either a single year, a range of years in the form c(1880, 1900). If no value is specified, then for the "ssa" method it will use the period 1932 to 2012;acceptable years for the SSA method range from 1880 to 2012, but for years before 1930 the IPUMS method is probably more accurate. For the "ipums" method the default range is the period 1789 to 1930, which is also the range of acceptable years. For the "napp" method the default range is the period 1758 to 1910, which is also the range of acceptable years. If a year or range of years is specified, then the names will be looked up for that period.

method - This value determines the data set that is used to predict the gender of the name. The "ssa" method looks up names based from the U.S. So-

cial Security Administration baby name data. (This method is based on an implementation by Cameron Blevins.) The ”ipums” method looks up names from the U.S. Census data in the Integrated Public Use Microdata Series. (This method was contributed by Ben Schmidt.) The ”kantrowitz” method uses the Kantrowitz corpus of male and female names. The ”genderize” method uses the Genderize.io ¡http://genderize.io/¿ API, which is based on ”user profiles across major social networks.” The ”demo” method is uses the top 100 names in the SSA method; it is provided only for demonstration purposes when the genderdata package is not installed and it is not suitable for research purposes.
countries - The countries for which datasets are being used. For the ”ssa” and ”ipums” methods, the only valid option is ”United States” which will be assumed if no argument is specified. For the ”napp” method, you may specify a character vector with any of the following countries: ”Canada”, ”United Kingdom”, ”Germany”, ”Iceland”, ”Norway”, ”Sweden”. For the ”kantrowitz” and ”genderize” methods, no country should be specified.

6 Additional Dimension to the Existing work

The challenges faced in each stage was analyzed and we found a unique way which help to derive the community detection of people based on their LastName.
In many countries people LastName does not have an actual meaning but they represent the place of the community people from their ancestors origin.
If we build this as an API we can allow any user to use this and based on the Last Name, we can train a model using Machine Learning and results can be sent as a response which gives the community place.
Consider Education enrollment of international students, Based on that we can perform analysis which can report which community people concentrate towards a Educational Concentration. Interesting patterns can be detected.

7 Results

Topic : Obamacare, Affordable Care Act

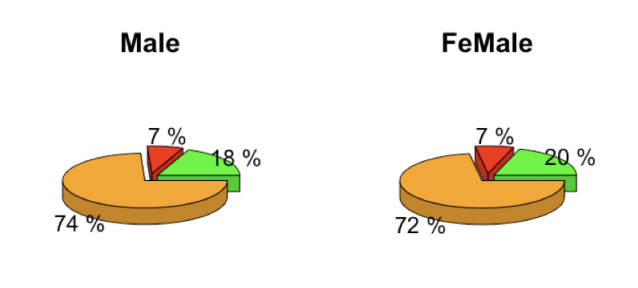

Figure 5: Atlanta - NewYork

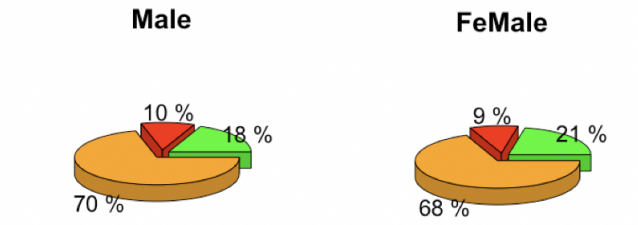

Figure 6: Atlanta - NewYork

8 Prospective tools, APIs and software

The following are the tools, APIs or the software which we are used for analysis:
i. Twitter API using R.
ii. Sentiment140 API for polarity based classification.
iii. Bulk Classification Service is used.
iv. R Gender Package is used.

9 Conclusion

Our Proof of Concept helps to find a correlation with the data extracted based on the polarity and the trending topic in it in various location.

10 Acknowledgement

We would like to thank Dr.Cao for his support for the project. We are able to narrow down the problem in more specific from his inputs for our work.

11 References

[1] Mondher Bouazizi and Tomoaki Ohtsuki. Sentiment Analysis in Twitter: From Classification to Quantification of Sentiments within Tweets. 2016.
[2] Nimita Mangal, Rajdeep Niyogi, and Alfredo Milani. Analysis of Users Interest Based on Tweets.
[3] Hassan Saif, Miriam Fernandez, Yulan He and

Harith Alani Evaluation Datasets for Twitter Sentiment Analysis A survey and a new dataset, the STS-Gold. 2013.

[4] Alec Go, Richa Bhayani, Lei Huang. Twitter Sentiment Classification using Distant Supervision. 2009.

[5] Kathy Lee, Diana Palsetia, Ramanathan Narayanan, Md. Mostofa Ali Patwary, Ankit Agrawal, and Alok Choudhary. Twitter Trending Topic Classification. 2011 11th IEEE International Conference on Data Mining Workshops

[6] Polarity Classification: http://help.sentiment140.com/api

[7] Gender Identification: https://cran.r-project.org/web/packages/gender/gender.pdf

[8] Twitter Geo parameter retrieval https://dev.twitter.com/streaming/overview/request-parameters