Exploration of Language on Twitter

Gregory Ceccarelli, Brennan Borlaug, Rajagopalan Mahadevan, Divyang Prateek

draft | **final**

MIDS W-251

Abstract

The rise of social media in the 21st century has given researchers a window into a communication from representing real-time stream of consciousness that documents thoughts and behaviors across human society. Popular platforms such as Facebook and Twitter make available, in large quantities, this data which is possible to parse and analyze with scalable open source technologies. Both the widespread adoption of these services and their arbitrary limits (e.g. Twitter's 140 character constraint) have caused a derivation of communication to form for nearly every language on Earth. Our goal is to build a prototype a platform that greatly simplifies the process of exploring the derived form of language found on social media platforms -- specifically Twitter.

Table of Contents

Introduction

As of July 2015, there were 2.3 billion active social media users (source: *We Are Social*) and growing. These users often exist on many different platforms at once and use them to fill specific roles in their lives. The social networking site Twitter has become popular for it's raw unfiltered timeline where users can go to view and discuss news, their thoughts and other content in real time. Given the speed at which Twitter moves, the large amount of content that passes through it, and the relative difficulty to track and analyze this traffic, we believed that the service could serve as a perfect test case for analysis.

Using archived Twitter data, we set out to explore how the construction, frequency, and sentiment of tweets vary across languages, time zones, and geographical regions. Specifically, we were interested in the ways in which a tweet's structure, content, and metadata varied across these groupings.

Additionally, we were interested and motivated to try and weave together a number of tools covered in this class. Our architecture, more fully described below, weaved together a number of open source tools including: Apache Cassandra for tweet storage, Apache Spark for some batch computation and R for additional computation and ultimately visualization. Python was selected as our "glue" language and is leveraged extensively.

Ultimately, we have presented the end product of our work as an interactive analysis in the form of a publicly available dashboard for external exploration. This dashboard contains a data sample of 500,000 tweets collected via Twitter's streaming API between 2015-2016 and serves as an effective proof of concept for a more ambitious and expansive platform in the future.

Motivation

The marketing landscape has been transformed with the introduction and proliferation of social media. It has opened up new avenues to reach potential customers and revenue streams that had not existed or were too complicated to tap into before. The data is available, but there lacks a simple way to extract, transform, and present this wealth of knowledge to promote impactful data based decision making in the business space. A research platform focusing on the ways in which people communicate online would be a boon to those looking to understand and connect with their target audiences on a more personal level.

Targeted advertisements and big data are key factors in many 21st century companies' digital strategies. Since 2013, the IAB Internet Advertising Revenue Report has shown that marketers now spend more on their Internet advertising budgets than on broadcast TV. With so many players competing for the limited attention of a population, it is critical that a company not only be spending their limited resources to target the right audience, but communicating with them in a way that maximizes effectiveness. By studying the construction and context of language online across different locations and demographics, marketers can obtain valuable insight into the people they are trying to reach and allow them to build an effective advertisement campaign around these insights. This analysis could extend to the level of human sentiment, the detection of sarcasm, or the prevalence of abbreviations (chatspeak) and emoticons in the conversations of targeted subpopulations.

The value of a research platform for the exploration of language on social media should not be restricted to just business and marketing domains however. Access to such a resource could provide a general pulse of the human population at any given time. This could be immensely valuable for disaster response or reactionary analysis to current events. Long-term sentiment analysis could be used to detect growing civil unrest before extreme action is taken or to detect the early signs of depression within certain communities.

Domain Challenges

Nearly every interaction on a social networking website is being collected and stored in some fashion. This includes all Foursquare check-ins, hundreds of Twitter tweets, and thousands of Facebook “likes” for each user on the platform. This data explosion can overwhelm many companies who are unprepared to deal with it’s sheer scale and became the primary challenge that we faced in designing a large-scale social media-based research platform. It is also the reason that our proof of concept is limited to a sample of Twitter tweets collected from 2015-2016.

Additional challenges faced during the design and implementation process included:

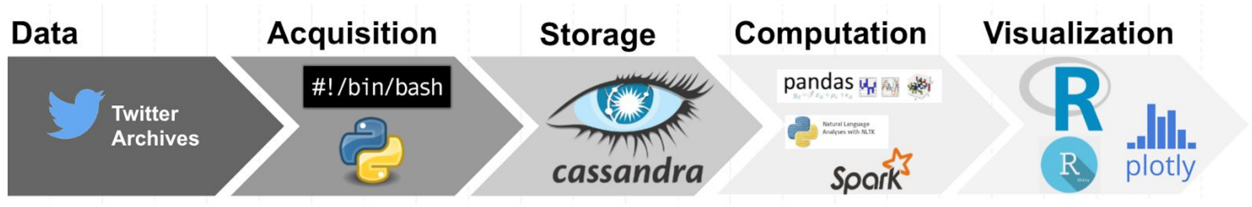
- Identifying and leveraging the most efficient methods to:
    - Create and manage a distributed infrastructure.
    - Collect, clean, and store tweet data.
    - Generate sentiment scores on diverse (multi-language) input data.
- Determining the best approach to model, report and visualize findings.
- Providing proper documentation for reproducibility and further extension.

Our Architectural Design

By leveraging open source big data technologies such as Apache Cassandra, Apache Spark, Python, and R, the platform’s proof of concept demonstrates a fully distributed processing pipeline presented via a front end Shiny UI. The architecture allows for maximum scalability and reusability.

Following the KISS principle, we leveraged a fairly straightforward architecture for our processing pipeline. Most of the components, at some point, were covered in class and we felt that reusing them for the final project would provide us a solid fundamental baseline in their nuances and complexities (as well integration). An overview of the process flow depicted in **Figure 1** is described below. Particular detail regarding the actual implementation of these tools for each section of work is covered in the subsequent section.

Figure 1: High Level Architecture



Process Flow

In general, at a high level, our architecture intends to do the following:

- Fetch and obtain data based on a pre-selected list of Twitter Archives from Archive.org
  - In batch download 3 of the selected files ~30-40 gb each compressed
  - Semi-uncompress them to disk on multiple nodes
  - Load them into a multi node cassandra cluster
- Build a set of representative tables (column families) use CQLSH that serve as log points for the data downloaded
  - Tweets -- store all relevant information from the JSON payload in cassandra
  - Tweets_ETL -- capture process meta data including: the file the data was run from, the time the process started/ended, how much data was read and how much data was written
- Do computation on the tweets stored in Cassandra using multiple toolsets
  - Use Spark / Python & Sentiment Analysis libraries to attempt to process the the tweet data in an effort to compare the sentiment of the tweet with emoticons present within
  - Use R to do similar computation on a broader scale. Generate more metadata about each tweet such as length, sentiment, emoticon count, type, etc used to visualize the data
- Visualize the data using R, Shiny Server and the Flexdashboard package

Tools and Technology

An enumerated listing of the scope and variety of tools and data used to implement the above architecture and processing pipeline is below.

Data

Sources: Twitter Archive Data, Sentiment Analysis Training Examples
Variety: Semi Structured Jason, Unstructured Flat Files, CSVs
Volume: 90-120 GB compressed in tar files, ~855 GB uncompressed
Velocity: Not relevant in our case as we completed most of the process in batch

Architecture Componentry

Storage: 9 node Cassandra Cluster
Processing: Apache Spark, Python, R, Bash Scripts
Acquisition: Python URL/FTP modules, Bash Scripts
Visualization: R, Shiny, ggplot2, plotly, flexdashboard

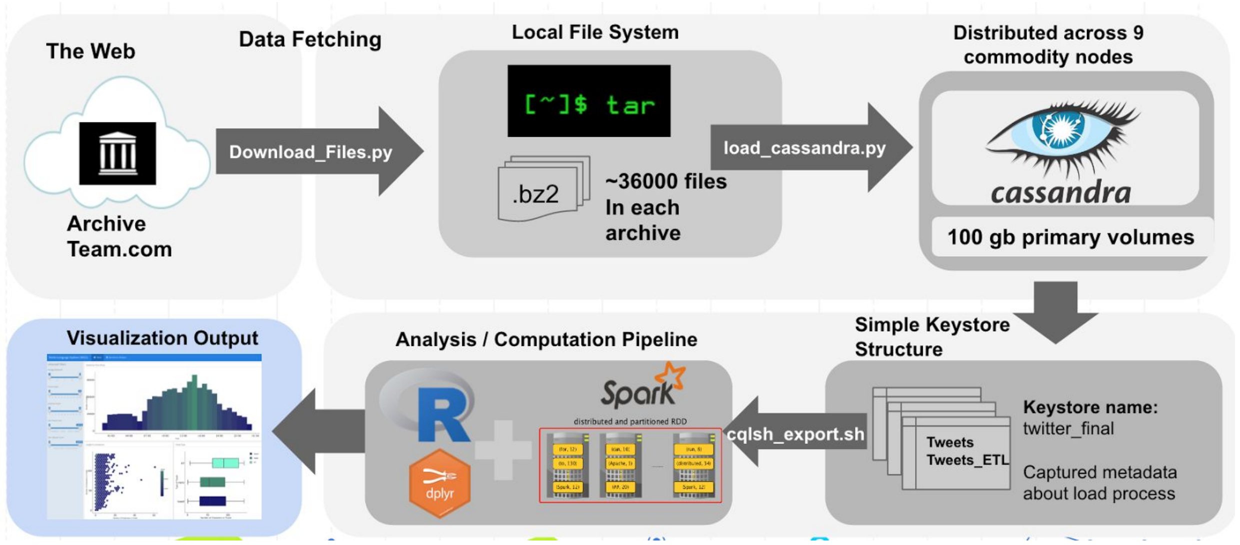
Hardware

For collection, processing and database storage we leveraged the Softlayer platform. We determined our application involves mid-size data processing and so attached a second 100GB disk to each node for persistent data storage. Since Cassandra functions well across multiple nodes and since Spark demands significant memory resources when running jobs, we figured the below configuration per node would be appropriate. Ultimately we provisioned 9 nodes with the below configuration and managed them in parallel using cluster ssh.

OS	vCPU	Mem (GiB)	Storage (GB)
Ubuntu 14.04	4	8	2 x 100

Implementation Detail

Figure 2: Granular Implementation Detail


The Web
Archive Team.com
Data Fetching
Download_Files.py
Local File System
[~]$ tar
.bz2
~36000 files In each archive
load_cassandra.py
Distributed across 9 commodity nodes
cassandra
100 gb primary volumes
Simple Keystore Structure
Keystore name: twitter_final
Tweets
Tweets_ETL
Captured metadata about load process
cqlsh_export.sh
Analysis / Computation Pipeline
Spark
distributed and partitioned RDD
dplyr
Visualization Output

The dataset was built from multiple tweet archives downloaded from the 2015-2016 Twitter Spritzer archives (https://archive.org/details/twitterstream). Spritzer data represents a 1% sample of all public tweets over a given time period and is free to access via Twitter's streaming API. Individual tweets were parsed and loaded in parallel into a 9-node Cassandra cluster for long-term storage. All told, the dataset measured 855 GB in size with a replication factor of two. The database was indexed on time zone, timestamp, and user id with a primary key established on a unique timestamp/user id combination.

Once the data was loaded, a combination of Apache Spark and CQLSH were used to conduct the analysis. Summary reports were generated using R and Python which were fed to a front end Shiny interface for visualization. Relevant derived attributes provided via web interface include: tweet sentiment (ranging from -1 to 1), emoticons per tweet, tweet length, type of tweet (original, reply, retweet), hashtags, and word count.

For sentiment analysis in Python, the VaderSentiment package was used to determine the sentiment of the text in a tweet. The package classifies the text as positive ("pos"), negative ("neg"), or neutral ("neu"). If there are both positive and negative sentiments expressed in a tweet it is classified as "SentimentNotClear".

In R, a different library was used for this purpose -- Professor Bing's opinion dictionary was used which contains ~6800 words encoded as either positive or negative.

Our visualization can be accessed in two places:

- http://192.155.215.11:3838/w251
- https://gregce.shinyapps.io/tweet_viz/

Learnings

Technical Challenges

We faced a number of technical challenges during implementation:

- Scaling Cassandra:
    - Originally we spun up a three node cluster but realized fairly early on that we'd need to add additional nodes to ensure we didn't run out of space
    - Adding the nodes was a tremendous challenge -- when we turned on the additional nodes, they failed to appropriately bootstrap (despite our darndest attempt to ensure all the cassandra.yaml files were correctly configured) and stream data
    - This ultimately required an attempt to restore from a backup taken prior to the addition of the other nodes

    - After that failed to work - the nodes ultimately did come online and had data but we were getting widely inconsistent results returned when querying - we decided to start from scratch… We wiped all of the data & logs and reloaded the data into the 9 node cluster
    - Lesson Learned: be very careful in pre-planning because it's not always trivial to scale a Cassandra cluster and can cause a bit of a headache if things go awry

- Python Driver issues
    - Once our 9 node cluster had sufficient data written to it, we attempted to use the same python driver to read from it.
    - Unfortunately, for us, this was riddled with problems -- and we ran into a memory leak issue (documented here on stack overflow) that caused our nodes to repeatedly crash when multiple client sessions attempted to read data from the cluster
    - Ultimately, with no easy solution in sight, we resorted to creating a script to iteratively extract a large sample of information from our cassandra cluster across various timezone and input files.

- Decoding and matching emoticons in Python and R
    - Different types of encodings present numerous problems for working with text data across
    - Additionally certain emojis are stored with code points beyond the normal UTF-8 range and thus are fairly non-intuitive to work with
    - We ultimately leveraged the

Other Challenges

The group faced a number of collaboration and team member non-performance issues as well.

- This particular group had members across each time zone in the united states which made
    - We tried as best as possible to rectify this by scheduling meetings on "free days" like weekends instead of weekdays where schedule clashes were all too frequent.
    - We also had a lively gmail thread which was used to ensure we were all in sync
- Additionally, our team had two members (originally we were a group of 6) who failed to participate with the group work and who were ultimately asked to leave the group.
    - The lesson here is that err on the side of smaller, rather than larger projects and to ensure that everyone has their incentives aligned from the start of a project like this

Further improvements

- Improving robustness of pipeline
    - While we were interested in using Cassandra, we erred on the side of simplicity and chose batch vs. streaming storage / computation
    - One could conceivably build a streaming solution that would offer this insight in real time
    - Additionally, Cassandra, in our experience might not be the

- Incorporating Machine Learning
    - Use more data from the tweets, say location, time, etc for prediction. Right now we are mainly focused on using the text from the tweet for analysis
    - Our current model is the most basic thing we could think of, there might be other better model that can be used (**but we didn't explore that direction since ML wasn't the focus of the project**)
    - Expand the corpus to include much more data
    - Focus on actually delineating between true and false positives in the sentiment analysis phase -- we spent a trivial amount of time here.

Conclusion

By providing a platform capable of extracting, transforming, and presenting large quantities of Twitter data, we have demonstrated an effective proof of concept that can be expanded upon to incorporate additional services and functionalities going forward. Ultimately, if time permitted, we'd be interested in extending this proof of concept based on the improvements documented above.

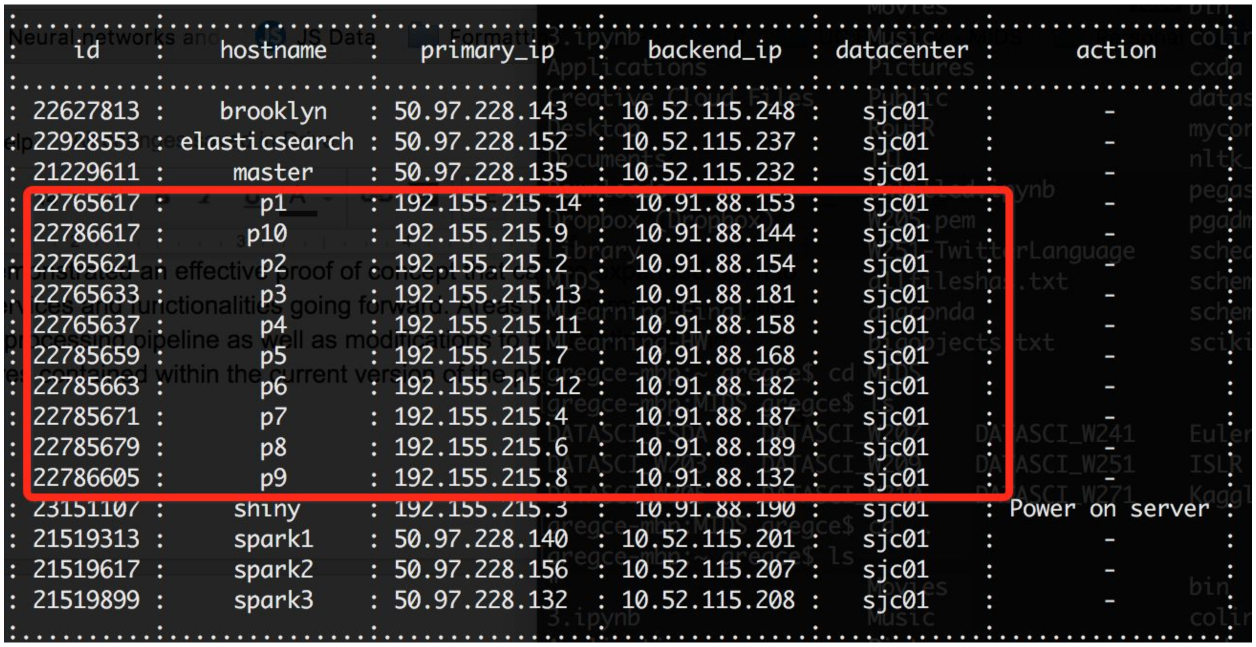
Appendix

Our Github Repository

We have a heavily documented github repository here which contains a breakdown of our project work. It is separated into the following sections. Rather than copy and paste all of the code here, We encourage the reader to investigate this resource.

- Configuration: Details instructions used to set up our 9 Ubuntu nodes
- Storage: Provides loading details + DDL for our cassandra key stores
- Exploration: Source code for data extraction and exploration in python and R
- Visualization: Final datafile + .Rmd file used to visualize our results
- Project Assets: Hard copies (pdfs) of our final project presentation, whitepaper (what you're reading) & screenshots of our running cluster

Additionally, here is a snapshot of our machine IPs. The master node was P1.



```
:..........:.................:.................:...............:............:...................:
:    id    :    hostname     :   primary_ip    :  backend_ip   : datacenter :      action       :
:..........:.................:.................:...............:............:...................:
: 22627813 :    brooklyn     : 50.97.228.143   : 10.52.115.248 :   sjc01    :         -         :
: 22928553 :  elasticsearch  : 50.97.228.152   : 10.52.115.237 :   sjc01    :         -         :
: 21229611 :     master      : 50.97.228.135   : 10.52.115.232 :   sjc01    :         -         :
: 22765617 :       p1        : 192.155.215.14  : 10.91.88.153  :   sjc01    :         -         :
: 22786617 :       p10       : 192.155.215.9   : 10.91.88.144  :   sjc01    :         -         :
: 22765621 :       p2        : 192.155.215.2   : 10.91.88.154  :   sjc01    :         -         :
: 22765633 :       p3        : 192.155.215.13  : 10.91.88.181  :   sjc01    :         -         :
: 22765637 :       p4        : 192.155.215.11  : 10.91.88.158  :   sjc01    :         -         :
: 22785659 :       p5        : 192.155.215.7   : 10.91.88.168  :   sjc01    :         -         :
: 22785663 :       p6        : 192.155.215.12  : 10.91.88.182  :   sjc01    :         -         :
: 22785671 :       p7        : 192.155.215.4   : 10.91.88.187  :   sjc01    :         -         :
: 22785679 :       p8        : 192.155.215.6   : 10.91.88.189  :   sjc01    :         -         :
: 22786605 :       p9        : 192.155.215.8   : 10.91.88.132  :   sjc01    :         -         :
: 23151107 :      shiny      : 192.155.215.3   : 10.91.88.190  :   sjc01    :  Power on server  :
: 21519313 :     spark1      : 50.97.228.140   : 10.52.115.201 :   sjc01    :         -         :
: 21519617 :     spark2      : 50.97.228.156   : 10.52.115.207 :   sjc01    :         -         :
: 21519899 :     spark3      : 50.97.228.132   : 10.52.115.208 :   sjc01    :         -         :
:..........:.................:.................:...............:............:...................:
```

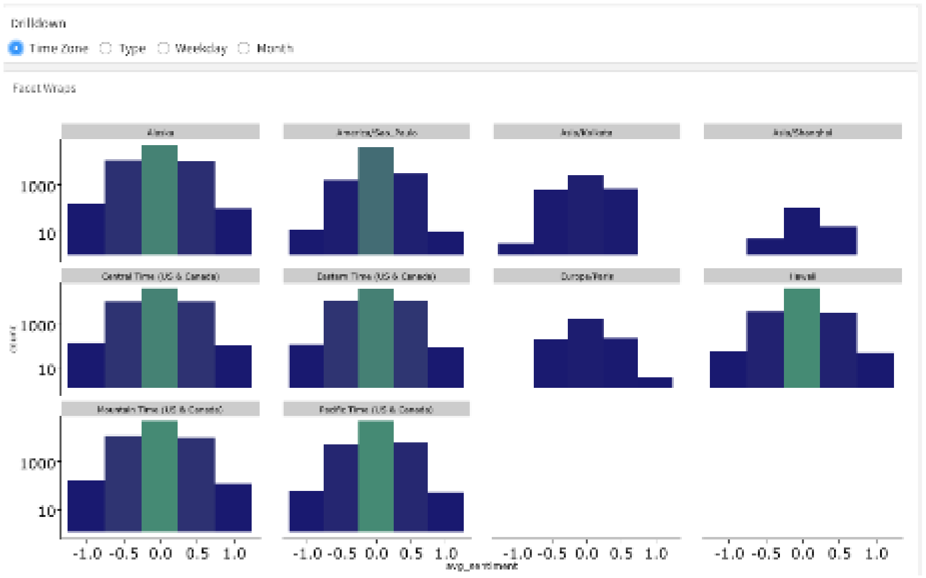
Sample Analysis: Tweet Sentiment

Goal: Document an exploration of the variation of tweet sentiment across different categories of users

Time Zone

The first grouping we explored was timezone/region. Here we used the timezone as a proxy for region. A majority of the time zones we explored were in the United States. This was primarily due to the location of this study. Some of the timezones we explored outside of USA included Paris, Kolkata, Shanghai and Sao Paulo.

An interesting finding while looking at the data with these categories was the amount of skew that was noticed in the sentiment in Asia and Europe, when we filtered out tweets that contained very few characters (Less than 30). In Europe, a majority of the sentiment was negative if not neutral. This could possibly be because of the financial and political distress prevalent in a majority of the continent. The refugee crisis, the war in Syria, the faltering economy in Greece and Spain, the terrorist attacks in France and Turkey all could possibly be compounding factors for this finding. In Asia on the other hand, the sentiment seemed overly positive. This could possibly be attributed to the booming economies of countries like India and China. There is also relative political and financial stability when compared to Europe. The United States by contrast was very balanced across all timezones including Hawaii and Alaska.

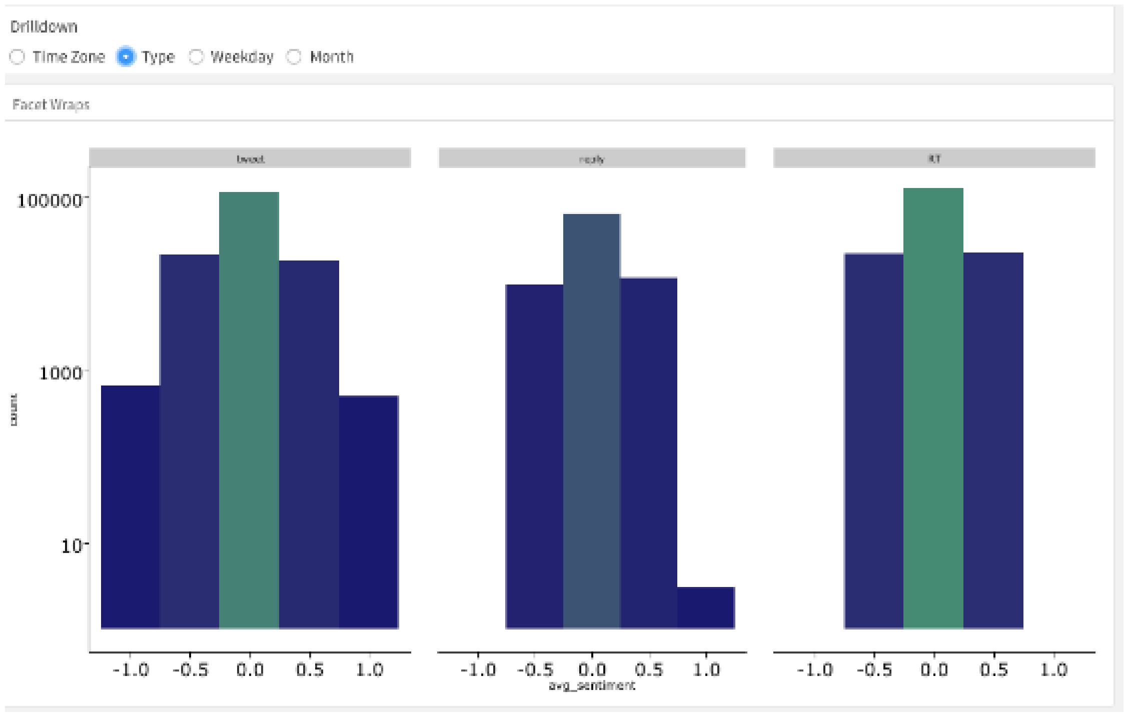


Sentiment analysis by Timezone/ Region (Min 30 Char)

Type of tweet (Tweet/RT/Reply)

When categorizing by the type of tweet, we noticed an equal balance in sentiment. This makes sense a frequent twitter users take to the platform for all ups and downs in the their lives. User

wish each other on birthdays, congratulate on achievements, express condolences on the loss of a loved one etc.
ReTweets or RT as they are popularly called, are similar in that there equal number of users that would retweet a ‘friend’s’ tweet irrespective of the sentiment. The only caveat happens to be that, overly negative or overly positive tweets don’t seem to get any retweets at all (Despite there being plenty of extreme sentiments in Tweet category). We are not sure if this had something to do with the library we were using for the sentiment analysis or it was because users are wary of re-tweeting extreme sentiments.
The third category, Reply happens to be positively skewed. It seems, more users tend to reply to tweets that that are expressing a positive sentiment. Something like “I just won a gold medal at the Rio Olympics” is likely to get more replies from other users than something like “I hate Rio! It’s so hot and humid here. ”


Drilldown

Time Zone [✓ Type] Weekday Month

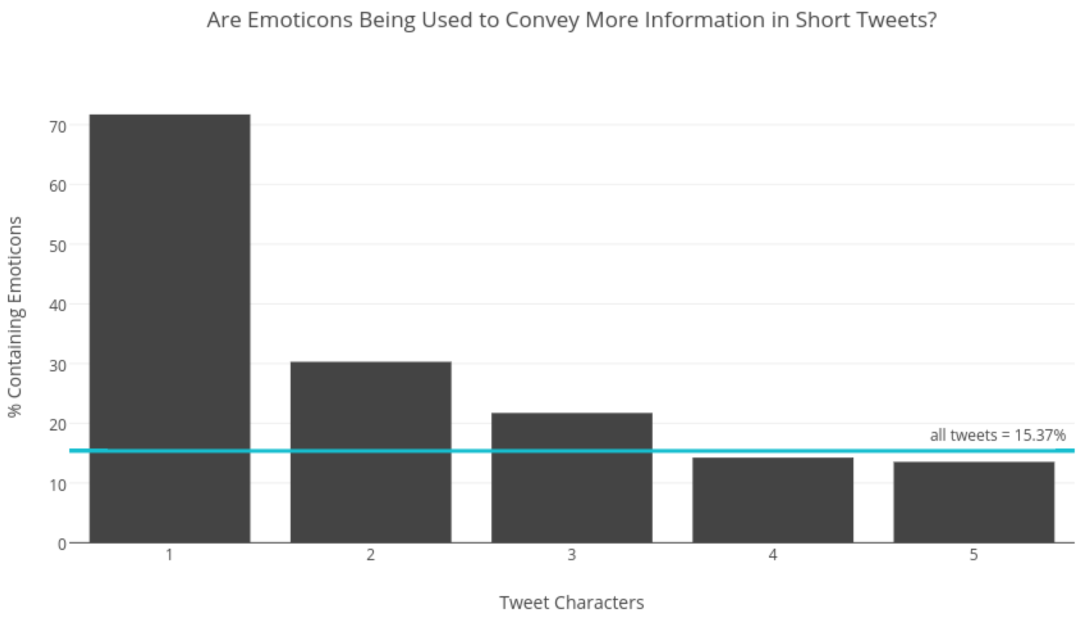
Sentiment analysis by type

Sample Analysis: Usage of Emoticons

Goal: Explore the tendencies of emoticon usage in tweets

An exploratory analysis into the tendencies of emoticon usage in tweets of short lengths was conducted to demonstrate a potential application for the web interface. The analysis aimed to discover whether users were more likely to include emoticons in tweets with less than five characters. An increased likelihood could suggest that users are including emoticons as a means express themselves more adequately than they could otherwise accomplish with a similar number of textual characters.

To conduct the analysis, a filter was applied to focus on tweets containing only one-to-five characters. For each length in this range, the proportion of tweets containing one or more emoticons was calculated and compared to the sample of all tweets in the dataset. It was found that an uncharacteristically large proportion of tweets of short length (especially those containing only one character) contained at least one emoticon. This result suggests that users are including emoticons in short length tweets for added context and expression that may not have been possible with text.


Are Emoticons Being Used to Convey More Information in Short Tweets?
% Containing Emoticons
all tweets = 15.37%
Tweet Characters