Exploring Human Language

Using social media data and scalable open source tools

Final Project Presentation
DATASCI W251: Scaling Up
August 21st, 2016

Greg | Brennan | Mahadevan | Divyang

Introduction

Social Media

Twitter is a microblogging platform where users generate huge amounts of textual data everyday in the the form of structured ***140 character tweets.***

Scalable Technologies

… like: **Cassandra** is a distributed keystore designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure

Research Potential

- A unique opportunity to:
  - Build on research that says Twitter is changing the way we speak and write (example)
  - Leverage open source big data technologies such as Apache Cassandra, Spark, R and Shiny
  - Demonstrate a full distributed processing pipeline, analysis and visualization using R and Shiny

Motivation

Guiding Hypotheses

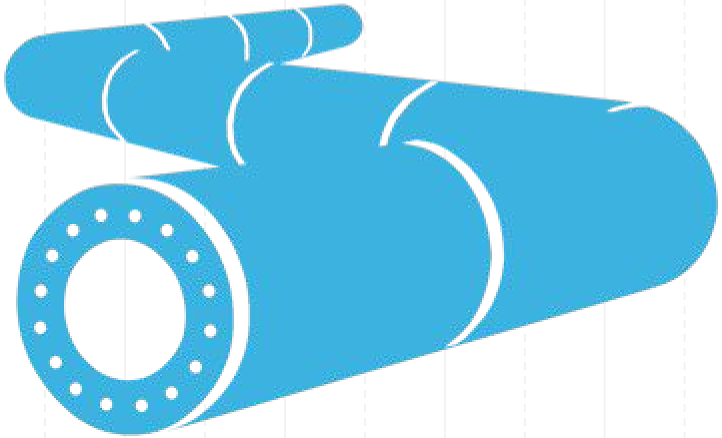
- Emojis and picture based forms of communication are beginning to dominate the way we communicate
- Different regions have different usage patterns on Twitter which likely represent the underlying communication patterns of the people using the service

Applicability

- Huge twitter data volume is a ripe source for automated sentiment analysis
- Thoughts tend to be expressed very vocally by users on Twitter
  - These thoughts can be reduced to positive / negative sentiment scores
- Past research attempts stopped short of producing reusable systems at scale
- Allows for the exploration of multiple technologies explored in w251

Project Goal

Collect, load, transform and analyze data from archived tweet streams to build a Proof of Concept system to explore the way people communicate, their sentiment and the rise of the emoticon



Problems to solve included:

- Identify and leverage the most efficient methods to:
  - Create and manage a distributed infrastructure
  - Collect, clean and store Twitter Data
  - Generate sentiment scores on multi-language, diverse input data
- Model, report and visualize
  - Volume of data collected, analyzed and final outcomes reported on
- Allow for reproducibility and further extension if so desired

Business & Societal Value

A platform exposing quantified characteristics of language(s) in near real time can have the following uses:

- Supports marketers attempting to launch campaigns specific to certain populations
- Enables a better understanding of real time events - political, disasters, sports outcomes, etc
- Aids comparative linguistics research for the masses

Architectural Overview

Architecture / Process Overview

Data **Acquisition** **Storage** **Computation** **Visualization**

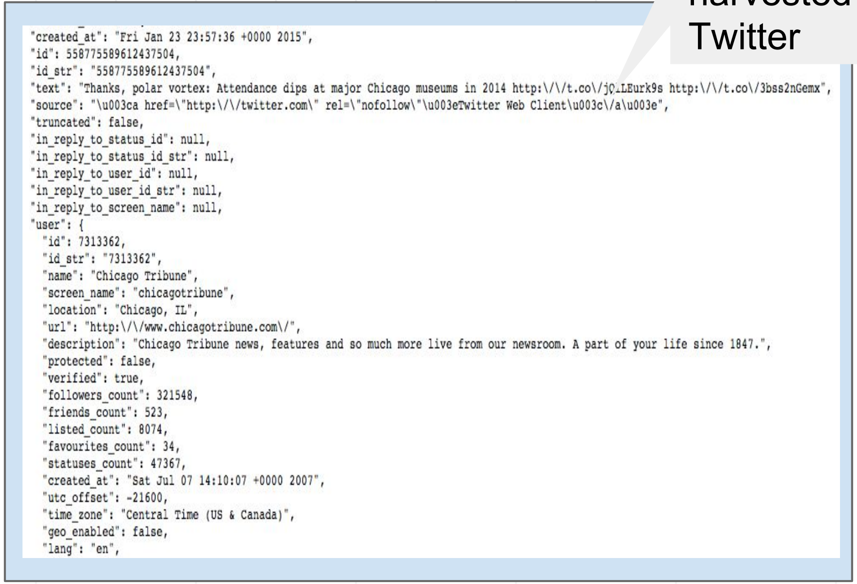
- Identify candidate Twitter Archives from the Twitter Spitzer Archive
- Download multiple archives from 2015 & 2016
- Clean and standardize while processing with custom python

- Load ~855 gb uncompressed twitter data into a 9 node cassandra cluster
- Create redundant keyspace
- Define DDL for relevant Column Families -> Twitter Schema + ETL logging table
- Process with Apache Spark across

- Write bash scripts to extract samples for viz
- Extensive data cleaning and sentiment analysis processing with R lang.
- Visualization and Web App built with Plotly libraries, Shiny Server & Flexdashboard

Twitter Data

- Per Twitter, (https://dev.twitter.com/overview/api) “Tweets are the basic atomic building block of all things Twitter. Tweets, also known more generically as “status updates.”
- They are also the building block of most of our application
- Using techniques and tools covered in class, we downloaded the text from massive archives
- Apache Cassandra + Multiple Machines + Python was leveraged to do so (and more fully described throughout)



```
"created_at": "Fri Jan 23 23:57:36 +0000 2015",
"id": 558775589612437504,
"id_str": "558775589612437504",
"text": "Thanks, polar vortex: Attendance dips at major Chicago museums in 2014 http:\/\/t.co\/jQ[illegible]LEurk9s http:\/\/t.co\/3bss2nGemx",
"source": "\u003ca href=\"http:\/\/twitter.com\" rel=\"nofollow\"\u003eTwitter Web Client\u003c\/a\u003e",
"truncated": false,
"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": {
  "id": 7313362,
  "id_str": "7313362",
  "name": "Chicago Tribune",
  "screen_name": "chicagotribune",
  "location": "Chicago, IL",
  "url": "http:\/\/www.chicagotribune.com\/",
  "description": "Chicago Tribune news, features and so much more live from our newsroom. A part of your life since 1847.",
  "protected": false,
  "verified": true,
  "followers_count": 321548,
  "friends_count": 523,
  "listed_count": 8074,
  "favourites_count": 34,
  "statuses_count": 47367,
  "created_at": "Sat Jul 07 14:10:07 +0000 2007",
  "utc_offset": -21600,
  "time_zone": "Central Time (US & Canada)",
  "geo_enabled": false,
  "lang": "en",
```


Example JSON payload response harvested from Twitter

Sentiment Analysis Data

- Attempting to create positive and negative sentiment training data would have been a lengthy endeavor

- To alleviate the burden, we leveraged UIC professor Bing Liu's r opinion lexicon data, which has ~6800 words (including misspellings), each classified as either positive or negative.
    - 2041 positive words
    - 4818 negative words

accomplishment
accomplishments
accurate
accurately

Example of some positive word snippets

watered-down
wayward
weak
weaken

Example of some negative word snippets

Tools and Technology

- **Data Sources**
  - Twitter Archives
  - Sentiment Analysis Training Examples
- **Variety**
  - Semi Structured Compressed JSON
  - CSV
- **Volume**
  - ~855 GB total raw
  - ~160 GB actually loaded /compressed
- **Velocity**
  - Did not stream, was able to process and load

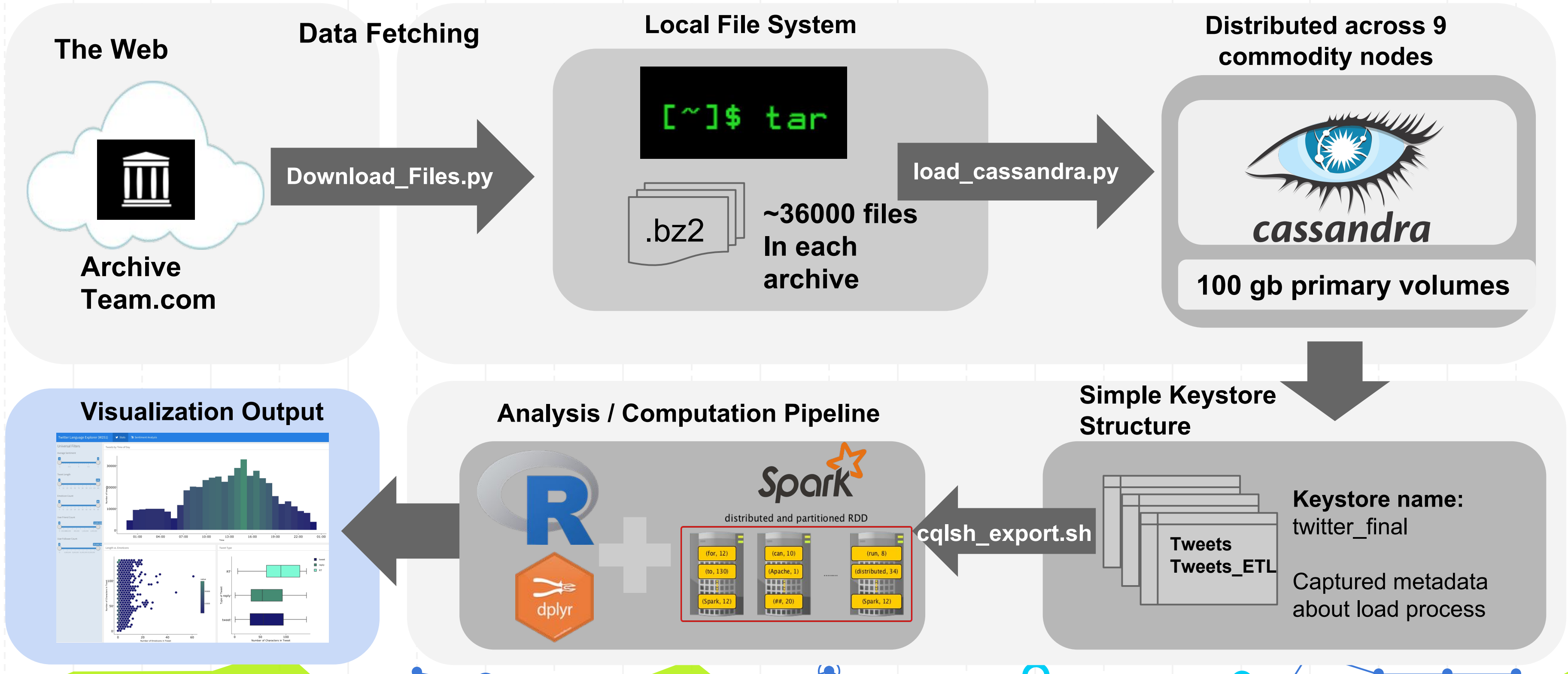
- **Data Acquisition**
  - Wget / Python / Bash / Tmux
- **Data Storage**
  - Cassandra
- **Data Processing**
  - Apache Spark
- **Computation / Analysis**
  - Natural Language Toolkit (nltk)
  - Data Analysis using Pandas, numpy (Anaconda)
  - R language, tidy text
- **Visualization**
  - Plotly
  - Shiny Server

- **Softlayer VM Configuration:**
  - 9 nodes in SJC
  - Type: Ubuntu OS
  - Memory: 8 GB
  - CPU: 4
  - Disk Space:
    - 100GB primary
    - 100 gb secondary

SOFTLAYER®
an IBM Company

Implementation

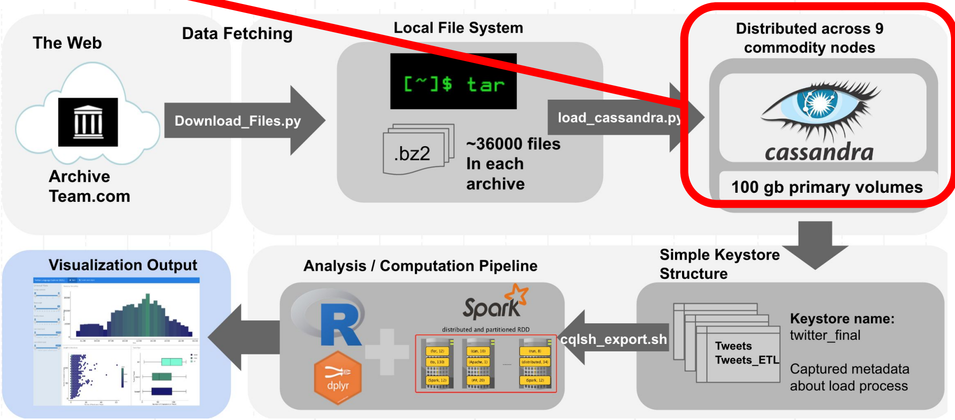
Data Acquisition & Storage Strategy



Machine Provisioning & Config

- Provisioned 9 machines in SJC using the Soft Layer command layer interface
  - ```
    slcli vs create --datacenter=sjc01
    --domain=gregceccarelli.com --hostname=p5
    --os=UBUNTU_14_64  --cpu=4 --memory=8192
    --disk=100 --disk=100 --billing=hourly
    ```
- Leveraged Cluster SSH to administer each machine in parallel.
- Installed relevant software including Java, Cassandra, Spark, Python (anaconda distribution), R, R server & Shiny Server
- Some nodes have specific tasks:
  - P1: ipython server
  - P4: Shiny server
  - P1, P2, P3: Cassandra seed nodes
  - P2, P3, P6: Parallel Loading nodes

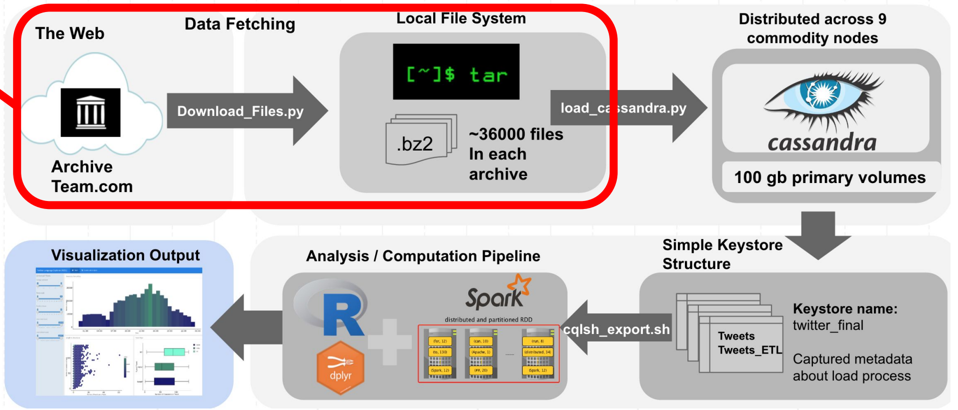
Relevant Piece of Architecture


    ```

Tweet Preprocessing

- Candidate archives were identified for downloading from https://archive.org/details/twitterstream
  - Ultimately, the following were downloaded:
    - **2016-06:** 43.2gb compressed, 345 gb uncompressed
    - **2016-09:** 30 gb compressed, 240 gb uncompressed
- A python/bash script was written to cycle through the files, download them and then untar them in appropriate /data partitions on the loading machines
- A python script was developed to iterate through each of the mostly uncompressed archives.
  - The script walks the directory,
  - Reads each .bz2 file into memory
  - Parses the contents of each of the file and identifies the relevant fields from the data

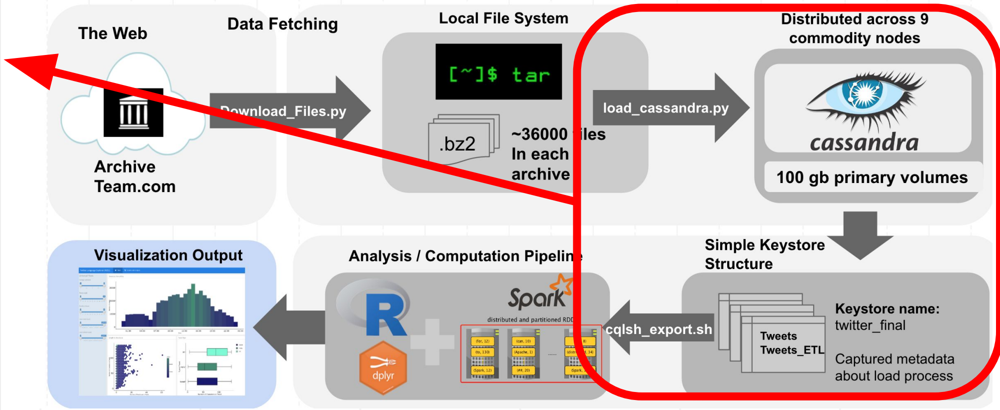
Relevant Piece of Architecture



Tweet Loading

- Well formed data, after read into memory, is used to construct a valid CQLsh statement.
- The cassandra-python driver is leveraged to interface directly with Cassandra -- each row in the input file is an individual insert into the database (woo, thank god for great write latency!)
- Our column family architecture is comprised of two relatively simple column families:
    - **Tweets:** Holds the main features of interest from the parsed tweets (e.g. id, user, text, timestamp,
        - Primary clustered key is: time_zone, user, timestamp → ultimately led to unbalanced nodes...
    - **Tweets_ETL:** Holds metadata about the loading process → took about ~24 hours to load all the data.

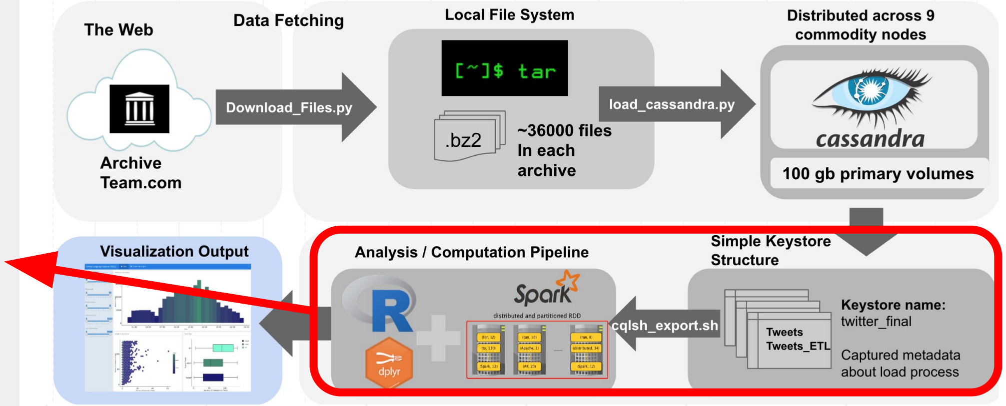
Relevant Piece of Architecture



Computation and Analysis

- Ultimately we conducted parallel processing of a large sample of the data stored in Cassandra (exported via a custom bash script due to driver connection issues) leveraging both Spark / Python & R
- **Spark / Python**:
  - Exploratory Analysis to determine whether emoticon presence matched text sentiment was conducted using Python , the [vaderSentiment]() library and Spark
- **R:** 30 sample extract files were read into memory (~4 gb approximately)
  - data was further cleaned to handle malformed strings
  - text of each tweet was decoded,
  - emoticons and language matched from custom dictionaries,
  - text was tokenized and sentiment was computed based on bing's dictionary

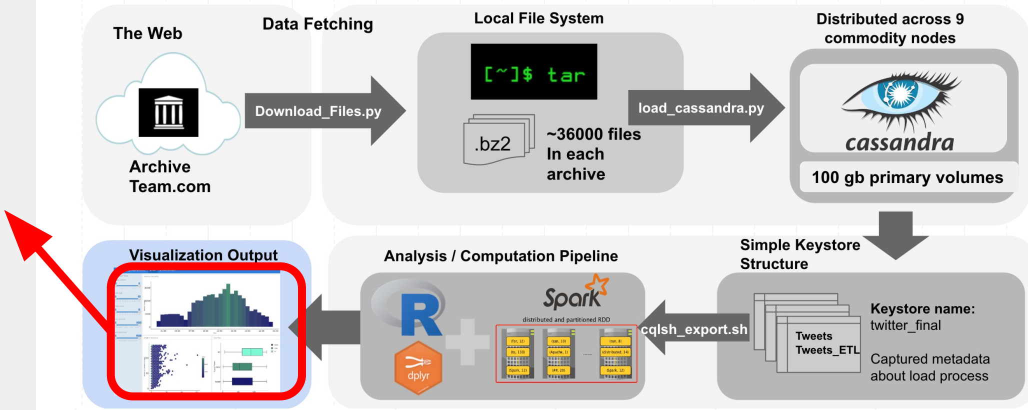
Relevant Piece of Architecture



Visualization

- R and associated libraries, which was used for final viz, leveraged a processed final table (data.frame) called: optimized_for_viz.rda
- Relevant derived attributes present after processing included:
  - Sentiment (ranging from -1 to 1) of tweet
  - Count of emoticons per tweet
  - Computed tweet character length
  - Type (tweet, reply, retweet)
  - Hashtag presence
  - Character length minus emoticons
  - Total words (fairly noisy)
- The dashboard was assembled using R shiny and a package called
- A shiny server was deployed to node P4 and the relevant files deployed to serve the viz

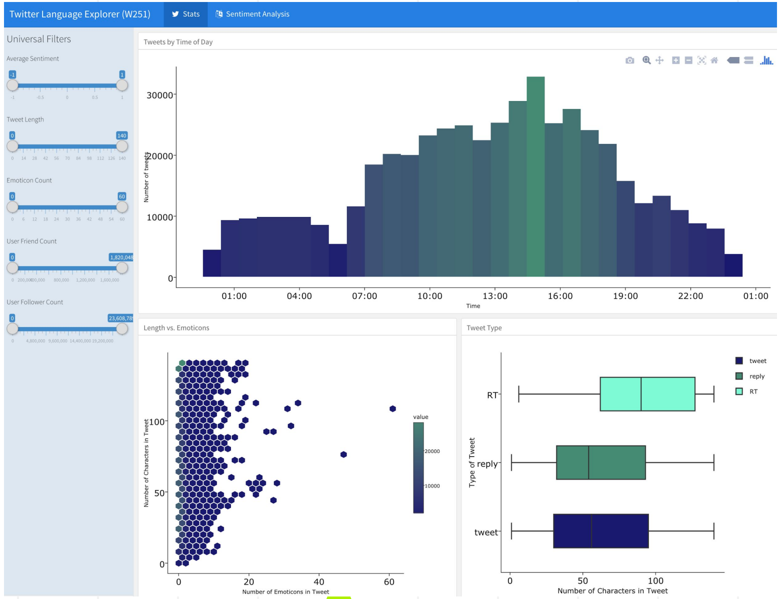
Relevant Piece of Architecture

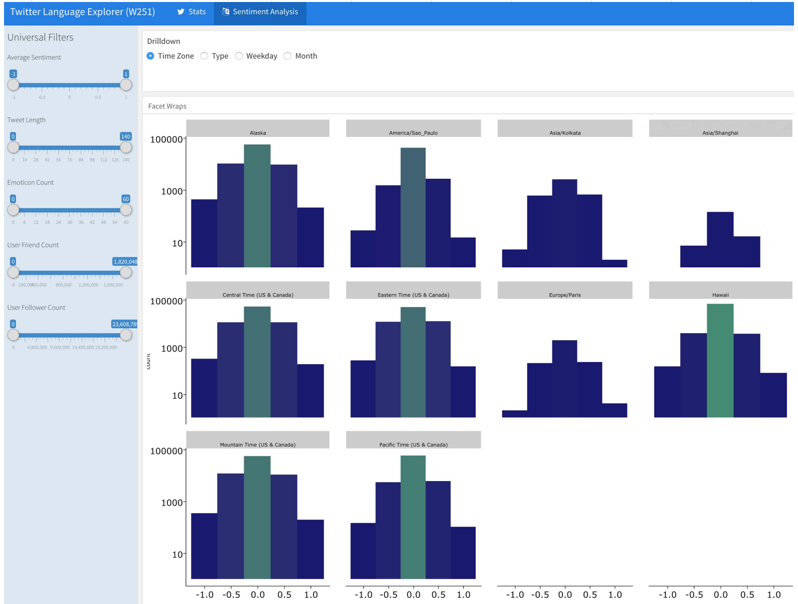


Results

Live Results

[Access here]





Summary

Hypotheses Recap

- Emojis and picture based forms of communication are beginning to dominate the way we communicate
- Different regions have different usage patterns on Twitter which likely represent the underlying communication patterns of the people using the service

Takeaways from E

- The vast majority of sampled tweets have one or less emoticon -- there is, however; a long tail.
  - Some observed tweets have up to 60 present emoticons
- There are definite times of day when people are more or less positive / negative
  - Positive Peaks: 4:30 pm, 6 pm
  - Negative Peaks: 3 pm
- The characteristics of tweet types tend to agree with our preconceived notions (retweets are longer in length than tweets)
- There are clear patterns that differentiate sentiment across time zones

Future Improvements and Learnings

Questions?

- A number of our processes are executed in batch or manually at the moment; these programs could be upgraded or modified to be run via more flexible or automated means
- We chose Apache Cassandra as a data store and it was intemperate at every turn. We ended up learning a lot but spent a ton of time bringing up dead nodes, monitoring the cluster and trying to diagnose bugs or problems
- Our focus wasn’t necessarily on uncovering the true answers to our hypotheses but rather building a pipeline to gather, transform and store this data -- much work could be done to engineer better exploratory models for this data
- We did not conduct heavy predictive or unsupervised learning -- but one could and should do this

Appendix / Resources

- [Twitter Language Explorer Viz]
- Our [white paper]
- Our [github repository]

Figure 1: Example Parallel Administration

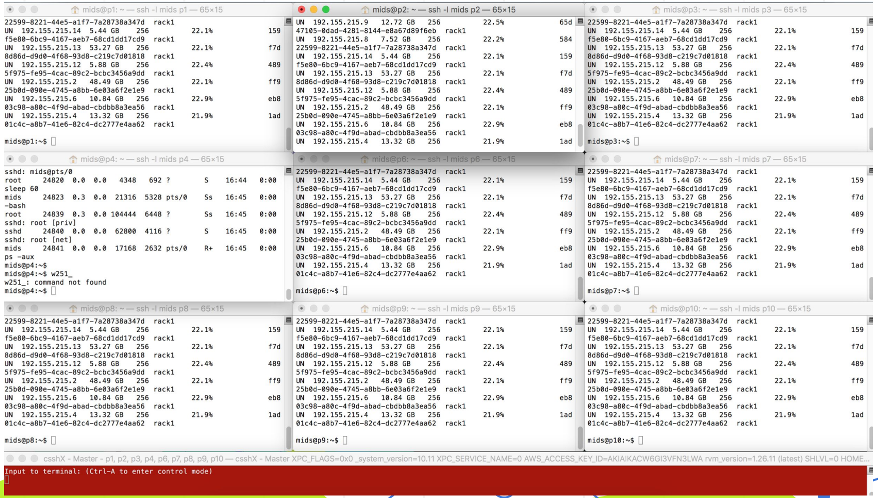
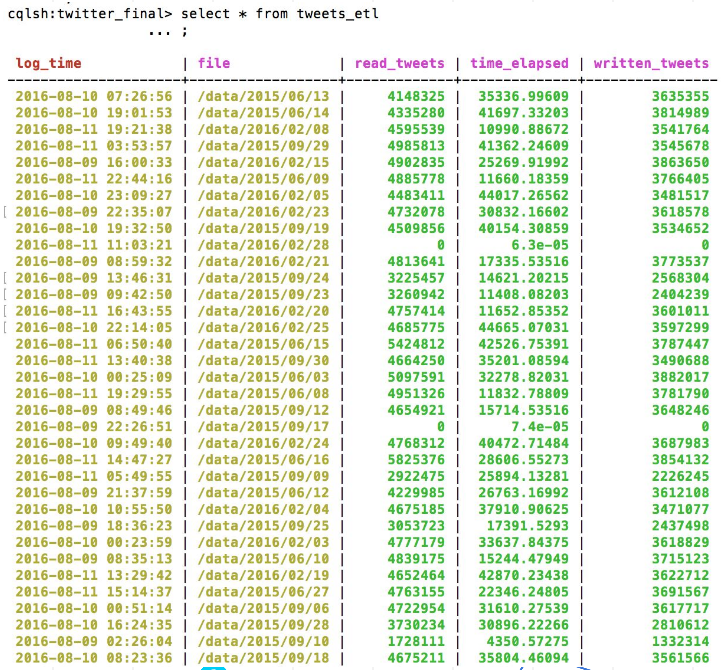


Figure 2: Tweet Load Stats



```
cqlsh:twitter_final> select * from tweets_etl
                 ... ;
```

log_time	file	read_tweets	time_elapsed	written_tweets
2016-08-10 07:26:56	/data/2015/06/13	4148325	35336.99609	3635355
2016-08-10 19:01:53	/data/2015/06/14	4335280	41697.33203	3814989
2016-08-11 19:21:38	/data/2016/02/08	4595539	10990.88672	3541764
2016-08-11 03:53:57	/data/2015/09/29	4985813	41362.24609	3545678
2016-08-09 16:00:33	/data/2016/02/15	4902835	25269.91992	3863650
2016-08-11 22:44:16	/data/2015/06/09	4885778	11660.18359	3766405
2016-08-10 23:09:27	/data/2016/02/05	4483411	44017.26562	3481517
2016-08-09 22:35:07	/data/2016/02/23	4732078	30832.16602	3618578
2016-08-10 19:32:50	/data/2015/09/19	4509856	40154.30859	3534652
2016-08-11 11:03:21	/data/2016/02/28	0	6.3e-05	0
2016-08-09 08:59:32	/data/2016/02/21	4813641	17335.53516	3773537
2016-08-09 13:46:31	/data/2015/09/24	3225457	14621.20215	2568304
2016-08-09 09:42:50	/data/2015/09/23	3260942	11408.08203	2404239
2016-08-11 16:43:55	/data/2016/02/20	4757414	11652.85352	3601011
2016-08-10 22:14:05	/data/2016/02/25	4685775	44665.07031	3597299
2016-08-11 06:50:40	/data/2015/06/15	5424812	42526.75391	3787447
2016-08-11 13:40:38	/data/2015/09/30	4664250	35201.08594	3490688
2016-08-10 00:25:09	/data/2015/06/03	5097591	32278.82031	3882017
2016-08-11 19:29:55	/data/2015/06/08	4951326	11832.78809	3781790
2016-08-09 08:49:46	/data/2015/09/12	4654921	15714.53516	3648246
2016-08-09 22:26:51	/data/2015/09/17	0	7.4e-05	0
2016-08-10 09:49:40	/data/2016/02/24	4768312	40472.71484	3687983
2016-08-11 14:47:27	/data/2015/06/16	5825376	28606.55273	3854132
2016-08-11 05:49:55	/data/2015/09/09	2922475	25894.13281	2226245
2016-08-09 21:37:59	/data/2015/06/12	4229985	26763.16992	3612108
2016-08-10 10:55:50	/data/2016/02/04	4675185	37910.90625	3471077
2016-08-09 18:36:23	/data/2015/09/25	3053723	17391.5293	2437498
2016-08-10 00:23:59	/data/2016/02/03	4777179	33637.84375	3618829
2016-08-09 08:35:13	/data/2015/06/10	4839175	15244.47949	3715123
2016-08-11 13:29:42	/data/2016/02/19	4652464	42870.23438	3622712
2016-08-11 15:14:37	/data/2015/06/27	4763155	22346.24805	3691567
2016-08-10 00:51:14	/data/2015/09/06	4722954	31610.27539	3617717
2016-08-10 16:24:35	/data/2015/09/28	3730234	30896.22266	2810612
2016-08-09 02:26:04	/data/2015/09/10	1728111	4350.57275	1332314
2016-08-10 08:23:36	/data/2015/09/18	4675211	35804.46094	3561566