Friday, Mar 2: Initial Proposal

Background and Motivation

One of the interesting facets of social networks is the interconnectedness of people, news, and geography. For marketers, politicians, and really anyone seeking to gauge public opinion on a certain topic, especially, Twitter, the live update forum, offers a window into such information. Unfortunately, it is in a non-aggregated format, much more conducive to individual monitoring. In this visualization, we hope to be able to present a global map that identifies twitter hotspots - that is, various zones will be depicted corresponding to regions of tweets that match user input.

Domain Goals & Audience

The domain goals comprise (1) visualizing the geographical distribution of tweets that contains certain words or phrases and (2) aggregating those data in a time-sensitive format [that is, query tweets in the last minute, hour, month, and year]. The intended audience is the general public, but more importantly those that would stand to gain from such public data, namely marketers or politicians.

Data

"Data: Tweet geographical location; Aggregate number of tweets; Time tweeted; words contained within tweet

The data will be conveyed primarily through a map visualization, with a user input box that queries all tweets within a certain period of time that contain the inputted words; hot spots will then be projected onto the map, corresponding to geographical location and frequency.

Where/how: Twitter api; googlemaps api; simply querying, storing into a temporary database"

Analysis Tasks

"Data retrieval: Where are the tweets from, and what is the respective frequency per location.

Correlation/Correlation: Do the tweets necessarily correspond to a geographic location, especially if the tweet contains a news item

Filtering: By time, one can visualize how the tweet distribution changes.

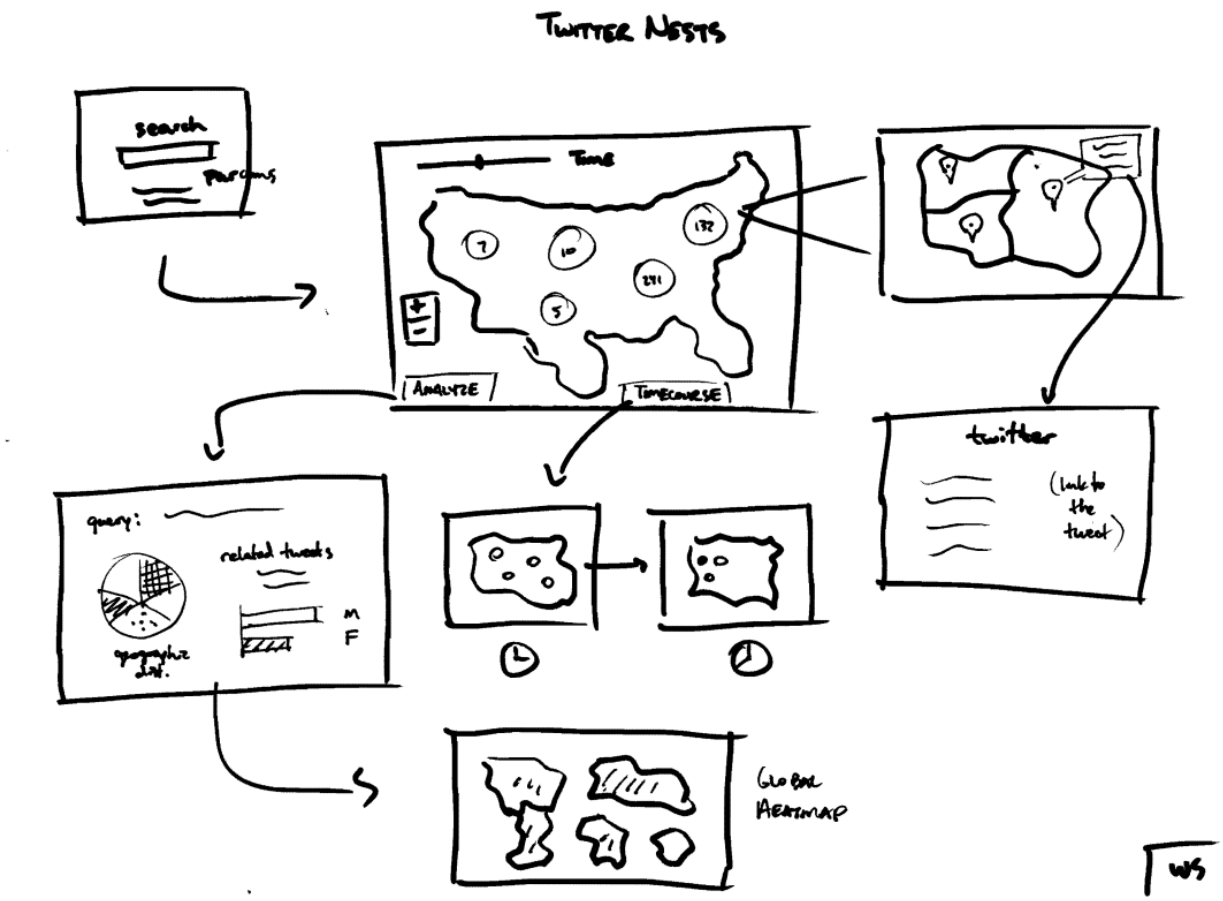
Aggregation and filtering will occur through simple user-interface that extracts only the relevant data from the temporary data table (from the initial query)."

Design Overview

The foundation of our visualization will be a heatmap of the world, with tweet densities visually encoded. Built into the map will be multiple filter and navigation functions to allow for user interaction, and we also hope to provide additional analytics for each query through graphs and charts on a separate page. Will has submitted our preliminary sketches.

Planned Schedule

We will try to break up the tasks, initially with two focusing on the back-end extraction and the other focusing on the front-end visualization of data. We plan to complete an alpha version in the next two weeks, and will work on enabling user interface and adding visual details (within tweet aggregates, such as zoom enabling) in the following weeks.



Sunday, Mar 4
Name was change from "TwitterNests" to "Tweetology"

Thursday, Mar 8: Meeting with Azalea
Discussed plan moving forward

- Google Maps v. Processing, what components
- How to handle Twitter API: pre-fetched results stored in database or real-time querying
- Platform: website (HTML5, CSS) or Java applet

Friday, Mar 9: Revised Proposal

Title

Tweetology (the study of tweets)

Sachin Patel, shpatel@college.harvard.edu
Will Sun, wsun@college.harvard.edu

Background and Motivation

One of the interesting facets of social networks is the interconnectedness of people, news, and geography. For marketers, politicians, and really anyone seeking to gauge public opinion on a certain topic, especially, Twitter, the live update forum, offers a window into such information. Unfortunately, it is in a non-aggregated format, much more conducive to individual monitoring. In this visualization, we hope to be able to present a global map that identifies twitter hotspots - that is, various zones will be depicted

corresponding to regions of tweets that match user input.

Domain Goals & Audience
The domain goals comprise (1) visualizing the geographical distribution of tweets that contains certain words or phrases and (2) aggregating those data in a time-sensitive format [that is, query tweets in the last minute, hour, month, and year]. The intended audience is the general public, but more importantly those that would stand to gain from such public data, namely marketers or politicians.

Data
Tweet geographical location; Aggregate number of tweets; Time tweeted; words contained within tweet

The data will be conveyed primarily through a map visualization, with a user input box that queries all tweets within a certain period of time that contain the inputted words; hot spots will then be projected onto the map, corresponding to geographical location and frequency.

Where/how: Twitter api; googlemaps api; simply querying, storing into a temporary database

Depending how efficiently we can query the Twitter API, we may decide to pre-fill a database from which we can analyze tweets. We also deiscussed potential difficulties with correlating location-based information from Twitter (ie. user-inputted home locations) with lat/longs on a map. The biggest challenge moving forward will be deciding how to handle the API data as well as to decide *what subset* of data is displayed (surely it won't be all tweets with "hello" in them if a user searches "hello).

We must also decide what platform to implement our program on; whether we wish to use Google Maps as the basis of our key visualization and implement the rest in a website-style format with HTML5/js, or if we wish to solely use Processing and tabbed plots. After our discussion with Azalea, we're more inclined to use Google Maps, if only to remove some of the challenges associated with low-level details like zoom and panning, so that we may focus on how best to display data and allow user interaction. This will depend on our experience with the API as we build our alpha.

Analysis Tasks
Data retrieval: Where are the tweets from, and what is the respective frequency per location.

Correlation/Correlation: Do the tweets necessarily correspond to a geographic location, especially if the tweet contains a news item

Filtering: By time, one can visualize how the tweet distribution changes.

Aggregation and filtering will occur through simple user-interface that extracts only the relevant data from the temporary data table (from the initial query).

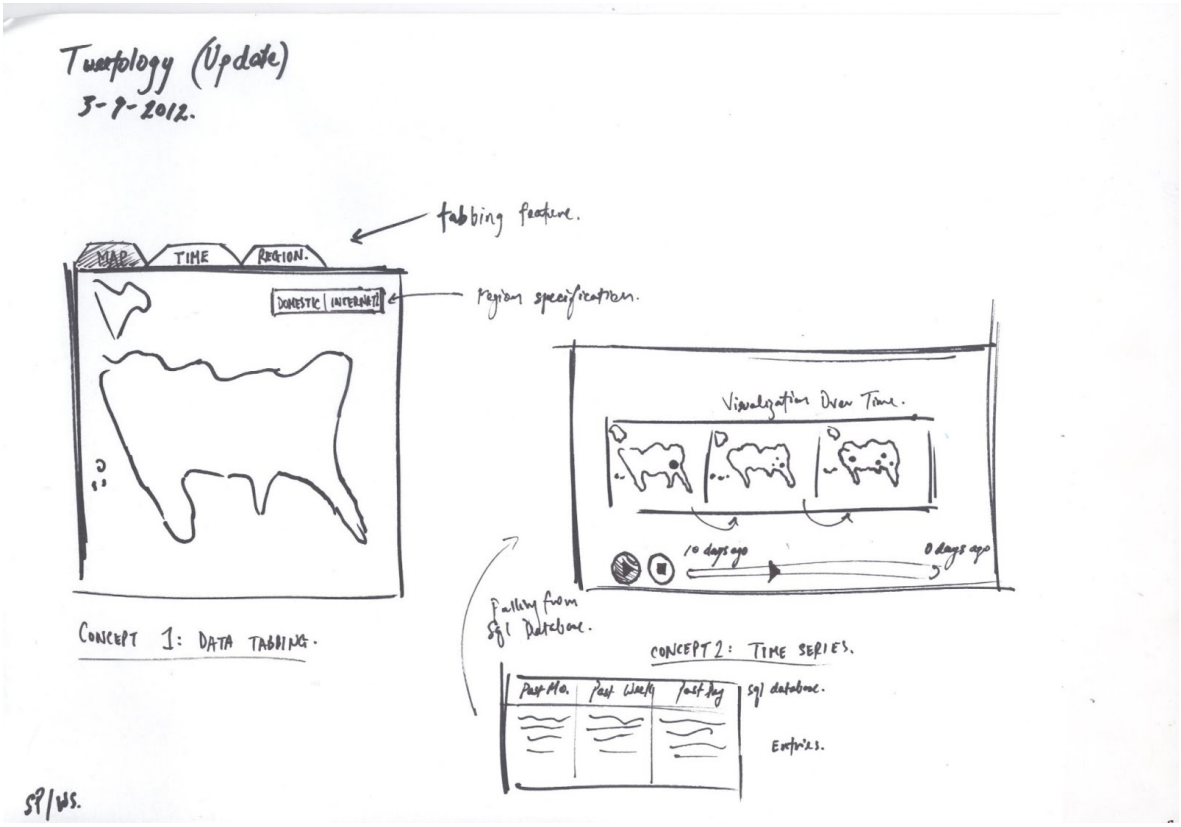
In addition, we also would like to filter based on queries (full-text search, other attributes).

Design Overview
The foundation of our visualization will be a heatmap of the world, with tweet densities visually encoded. Built into the map will be multiple filter and navigation functions to allow for user interaction, and we also hope to provide additional analytics for each query through graphs and charts on a separate page. Will has submitted our preliminary sketches, and Sachin has provided some updated ideas.

Planned Schedule
We will try to break up the tasks, initially with one focusing on the back-end extraction and the other

focusing on the front-end visualization of data. We plan to complete an alpha version in the next two weeks (prior to the Mar 30 checkpoint), and will work on enabling user interface and adding visual details (within tweet aggregates, such as zoom enabling) in the following weeks. Our major goal moving forward to is to answer the large questions associated with implementation choices (Google Maps v. Processing, real-time v. stored data), and these will be dependent on our experiences with these choices in the alpha.

Friday, Mar 9: Updated Sketch



Updates: time series depiction, data tabbing

Friday, March 30, 2012:

Updates:

Sketches (see below)

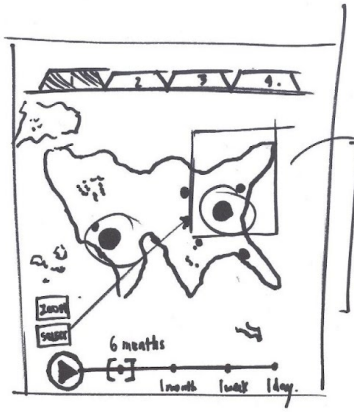
FINAL DESIGN : TWEETOLOGY.

* USING *RubyOnRails* for WEBSITE DEVELOPMENT.

① HOME PAGE

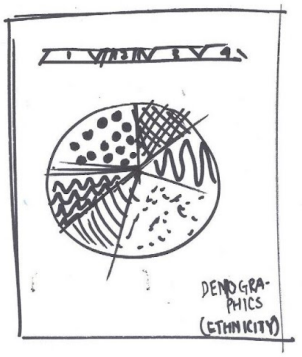
GET USER INFORMATION

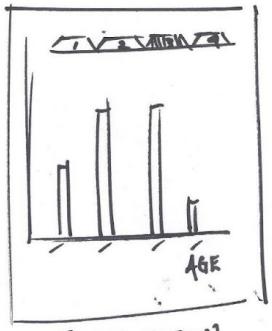
② VISUALIZATIONS. (PROCESSING).



OPTION TO CHOOSE REGION FOR REGIO-SPECIFIC VISUALIZATION & ZOOMING FUNCTION.

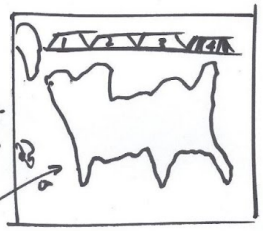
③ VISUALIZATIONS : CONTINUED.





IJEOMA MOTION GRAPH.

- COMPLEX VISUALIZATION.
- view multiple terms (different colors).



William Sun.
Sahar Patel
5/31/2012.

[Comments]

We decided to use framework that uses Ruby, specifically because its native interface including a specific Twitter API gem, which made querying tweets rather straightforward. In addition, we plan to use RoR or Sinatra to build our website, so such a decision naturally aligned with our method by which to scrape.

Here, we have listed some of the possible methods of accessing the twitter API that we considered:

- jSON object querying; javascript and php
- Ruby Twitter gem

In addition, another important facet moving forward was data storage. Again, we contemplated several design strategies, and specifically identified flaws with certain approaches:

- SqlLite Database through Processing: http://bezier.de/processing/libs/sql/
- Dynamically generate .tsv file after code is generated: one of the chief problems with this approach is if multiple users are using the interface, the *most recently* generated .tsv file is the one used as the datasource for the successive visualizations; we might alleviate this by generating timestamped tsv files

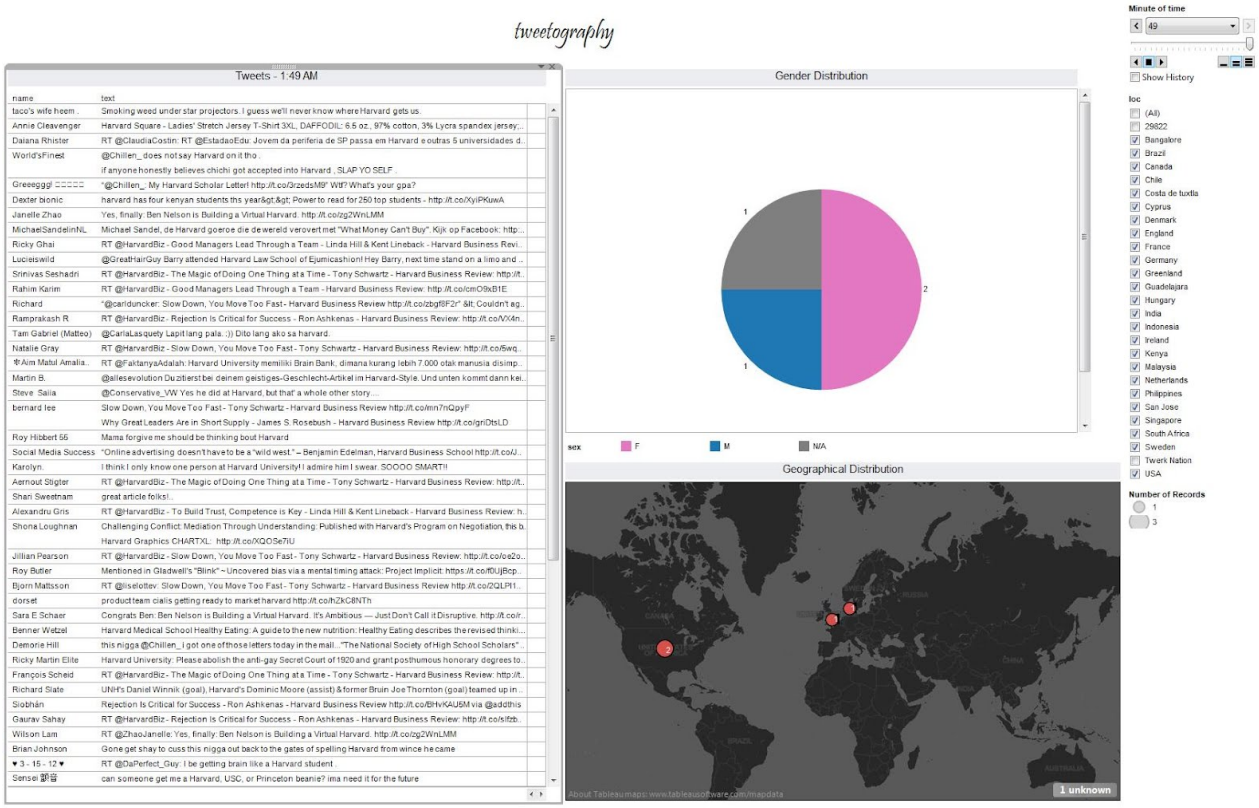
We are further exploring data storage strategies and intelligent design, and for the time being, are moving forward with the Sqlite Processing plug-in.

Will wrote a Ruby script that takes input of the form:

    ./extract.rb query {lat, lon, rad}

which searches for tweets using the "query" string and optionally, a "rad" mile radius around "lat" and "lon". This produces a csv file with tweet and user data which we further analyzed.

In addition, we have put together a brief visualization using Tableau, taking a small amount of Twitter data to gauge the integrity of our data acquisition process, the data density, and the graphical layouts of the final visualization. Below are the screenshots of the Tableau dashboard.



Tweetology

Sachin Patel and Will Sun

Introduction

One of the interesting facets of social networks is the interconnectedness of people, news, and geography. Indeed, rapid technological advancement has made it possible for millions around the world to tap into the vast landscape of information on the Internet. For marketers, politicians, and really anyone seeking to gauge public opinion on a certain topic, the live real-time update of Twitter offers a window into such information. Unfortunately, it is in a non-aggregated format, much more conducive to individual monitoring. In this visualization, we present a global map that identifies twitter hotspots - that is, various zones will be depicted corresponding to regions of tweets that match user input.

Facebook social network from December 2010

Our domain goals comprise of (1) visualizing the geographical distribution of tweets that contains certain words or phrases and (2) aggregating those data in a time-sensitive format [that is, query tweets in the last minute, hour, month, and year]. The intended audience is the general public, but more importantly those that would stand to gain from such public data, namely marketers or politicians. The novelty of this visualization predominantly lies in its capacity to aggregate tweets in real-time. Given the popularity of Twitter, many visualizations have been done with stored data sets - here we hope to help fill a gap in the field of social network visualizations.

Data

Twitter's API is arguably one of the best maintained sets of developer's tools on the market, making querying and acquisition of tweets in real-time possible. Their search API calls are extraordinarily powerful, allowing for basic keyword search as well as location filtering and other options. Additionally, the information returned on both individual tweets and users offer many possible data fields to study. Given that our primary goals rely on geographical and temporal information, Twitter's API more than satisfies our constraints.

Twitter recently introduced geocoding capabilities to its service, and subsequently incorporated this into their API as well. However, a large majority of users do not tweet with geo-enabled on - for a given query of 1500 tweets using 'Harvard' as a keyword, *only 15* tweets were geo-enabled, which provides precise latitude and longitude coordinates of the user's location. As we began to explore the data through command-line scripts, we quickly realized that this would be an issue. If the main feature of the visualization would be location-based, and only 1% of the data could be represented in this manner, the visualization would be significantly undercut in its effectiveness.

Fortunately, reverse geocoding APIs exist, including those offered by Yahoo and Google, allowing us to make fairly good estimates at a user's location based off of their user input. Though this would not be perfect (a user location might be "my room"), it increased the size of our location-enabled subset to over

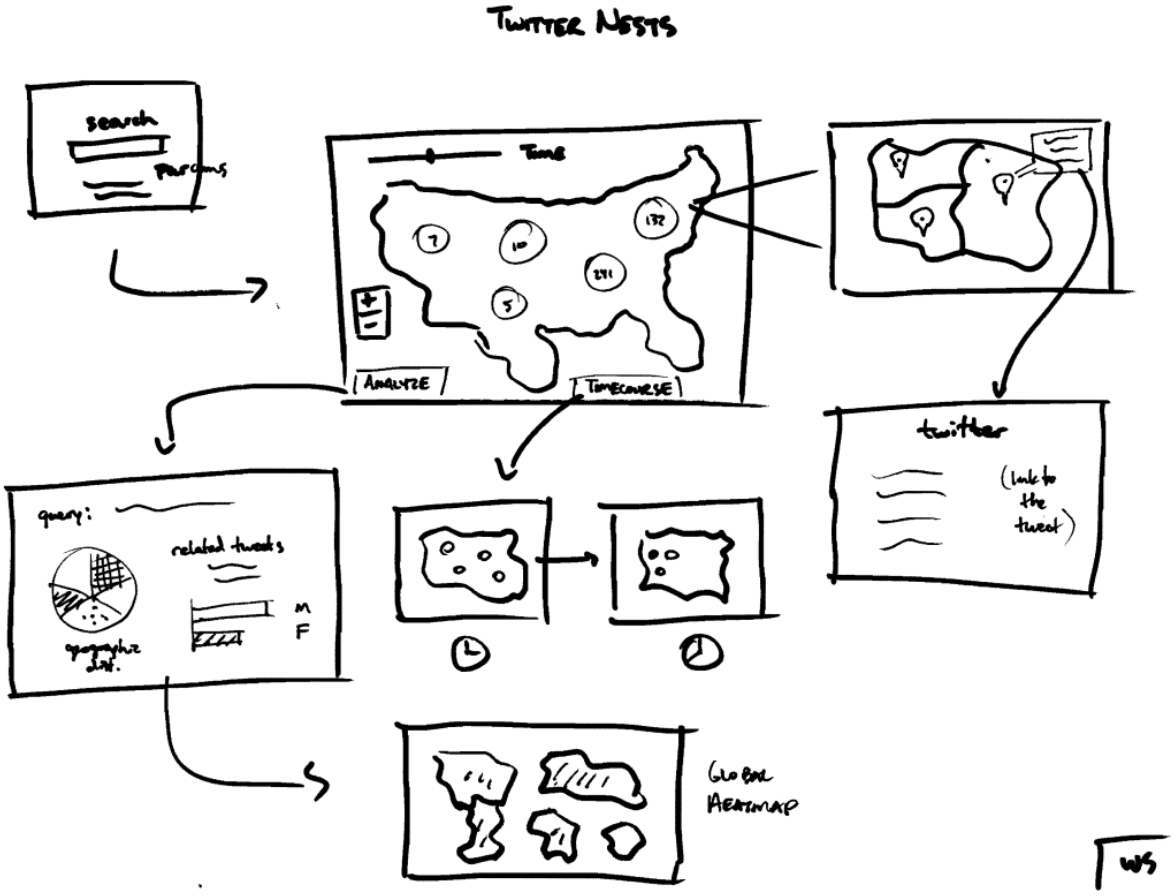
60% of the raw tweets returned.

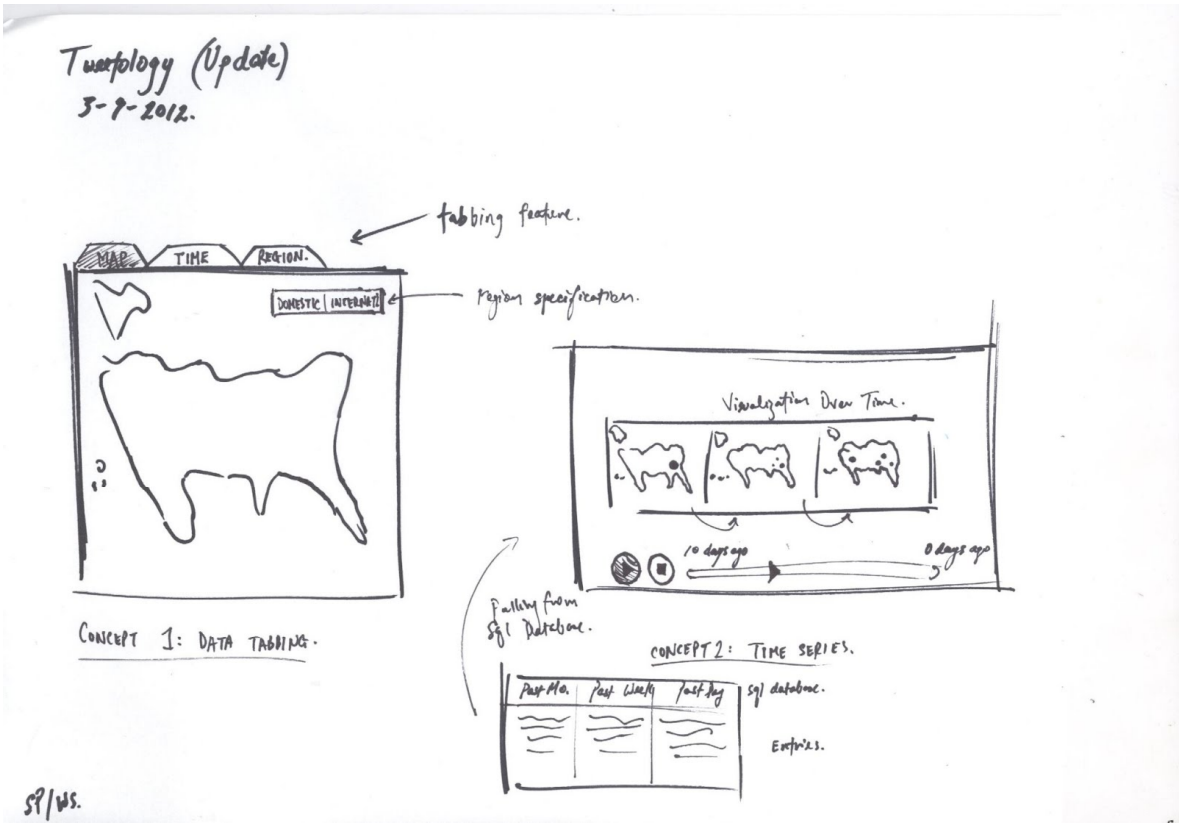
We ultimately were able to consider a wide variety of data fields, ranging from data on the user (number of followers), to linguistic analysis of the tweet (word cloud and mood), to geographical and temporal aspects of the tweeting.

Design

Early on, we decided that we would like to focus on such analysis tasks as data retrieval, correlation, filtering, and aggregation, since our primary goal was to provide a user-friendly interface for individuals to begin to make sense of the distribution of live tweets. We initially decided that the foundation of our visualization would be a heatmap of the world, with tweet densities visually encoded. Built into the map would be multiple filter and navigation functions to allow for user interaction, and ideally additional analytics for each query through graphs and charts on a separate page.

(preliminary sketches below)





However, after beginning to explore the data and the possibilities of what we could retrieve from the Twitter API, we realized that we would likely not have enough tweets for a given query to generate an actual global heat map; the densities would be insufficient. Below is a view of our initial data sets displayed in Tableau:

tweetography

Overall, the changes to our design throughout the process to our final product were predominantly driven by realizing both the possibilities and limitations of the technology frameworks we were dealing with, a process that no doubt is one that professional and amateur visualizers alike must go through.

Implementation

Back End

We chose Ruby on Rails as the framework of the project, with Processing as the main driver behind the visualization component. This was primarily due to our familiarity with both of these frameworks, but also from careful consideration of their capabilities relative to other alternatives. We considered using an API like Google Maps to handle the map in the visualization, primarily due to ease of use and not needing to handle lower-level tasks like zoom and filtering. With that strategy, we likely would have also used Google Charts or a similar graph platform. However, the downside to this approach was significantly less control and flexibility over what could be achieved. Processing allows for custom maps and graphs, and while Google Maps might have sufficed for some of our analysis tasks, Google Charts would definitely not. In considering whether to use Processing for the whole project, including data collection and parsing, we decided to use Ruby to handle some of the workload. Though Twitter API libraries exist for Java (ie. twitter4j), they are more unwieldy than the Ruby counterparts (Ruby Twitter gem). Moreover, the excellent resources we found for reverse geocoding (Ruby Geocoder) and mood determination (Twitter Sentiments) both were compatible with Ruby. It should be noted that Ruby Geocoder is an API that examines multiple geocoding APIs, including Yahoo and Google, so it was exceptionally powerful in returning potential locations for non-geocoded tweets/users.

There were tradeoffs that we had to make in pursuing this route, however. Though contact with the APIs was streamlined by using Ruby, this also meant that we would have to pass in the data to the Processing visualization. Processing does not have very much database support - incidentally, we did find a sqlite plugin that was later used in the storage of the map tiles. A workaround that we decided to implement was the generation of uniquely-named csv files for each search query. This would preserve separation between different users, were they to visit the website at the same time. The downside is that writing csv files takes time and space.

Another roadblock that we had to deal with were the rate limits of the Twitter API. Without an authenticated OAuth account, a user can only make 150 calls to the API per hour, and with authentication, the threshold only goes up to 350. Given that we were interested in both tweets and data on the corresponding users, that meant that we would be making 15 calls per query to the search API (Twitter returns maximally 100 tweets per page, with up to 15 pages) and up to 1500 calls per query to the user API, if all tweets were made by distinct users. Needless to say, we were rate-limited many times in the course of development. Luckily, since we ultimately planned to deploy to Heroku (a free hosting platform and the standard for Rails apps), there was a plugin available called Apigee that allowed us to bypass these API limits.

Front End

After contemplating between various data storage packages and available APIs, we decided to use Processing entirely for the map presentation: significantly, we wanted to enable rapid panning and zoom functionality, two features that were built in the Processing-compatible *Unfolding* library and its

respective TileMill .mbtiles generator. In addition, a SqlLite library had to be used to afford data storage of the multiple map tiles, which cumulatively allow zoom on 6 consecutive levels. Hence, we ran into several problems exporting, since our total applet size was 200 MB.

With regards to additional libraries, a wordcram library was imported to facilitate the development of a word cloud, a feature which would allow a rapid yet distilled representation of a selected group of tweets. Moreover, the controlP5 library was appended to our sketch, mostly due to its relatively easy user interface and coding functionality (such characteristics allowed us to design buttons rather easily).

Final Design

With regards to the design, we completely abandoned the tab view format in favor of a more dynamic, interactive, and user-friendly two-page presentation, one that allowed the visualization of the entire twitter map on the first page and data inspection on the second. Presumably, we figured that the most excitement - at least visually and economically - would be to present the map in the first panel.

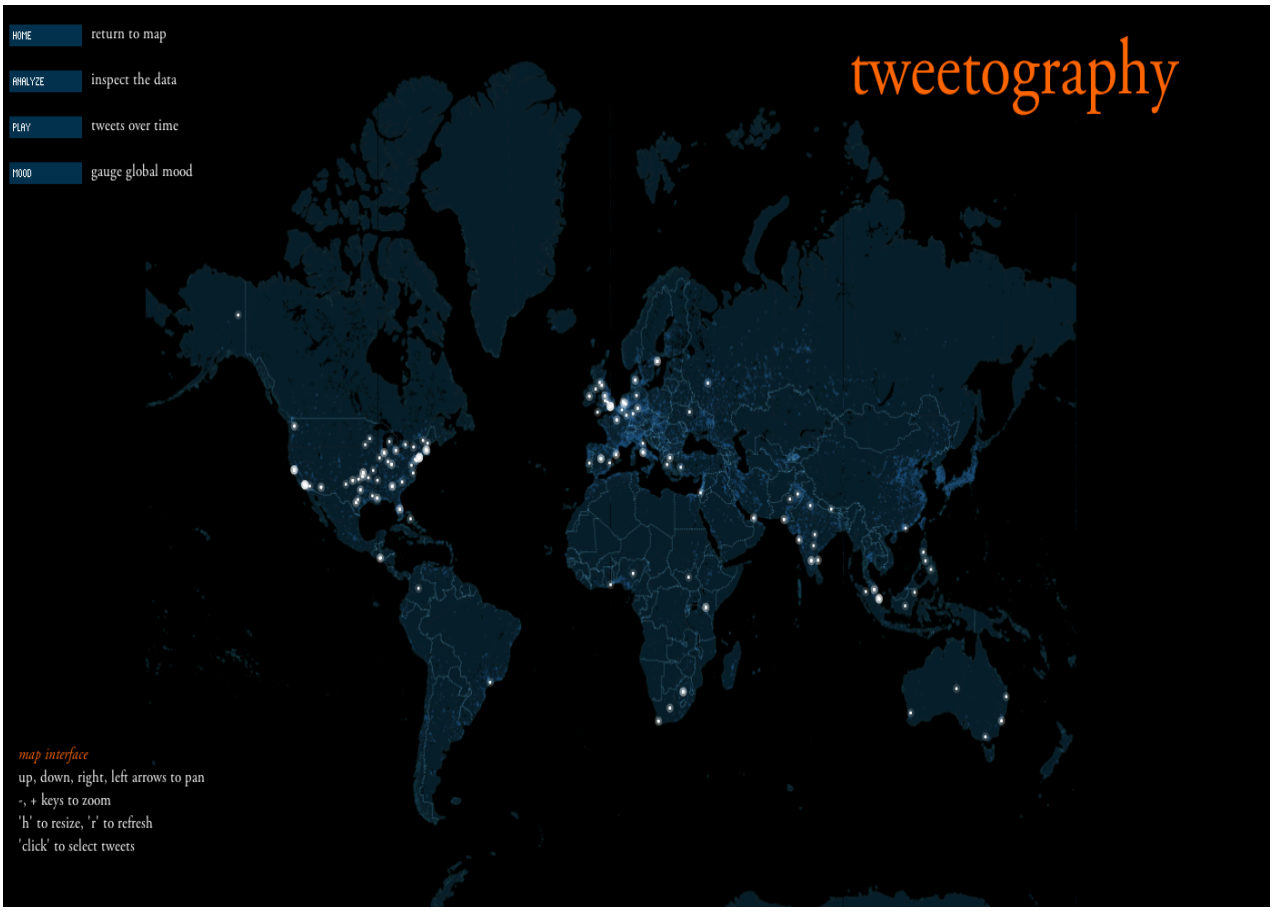
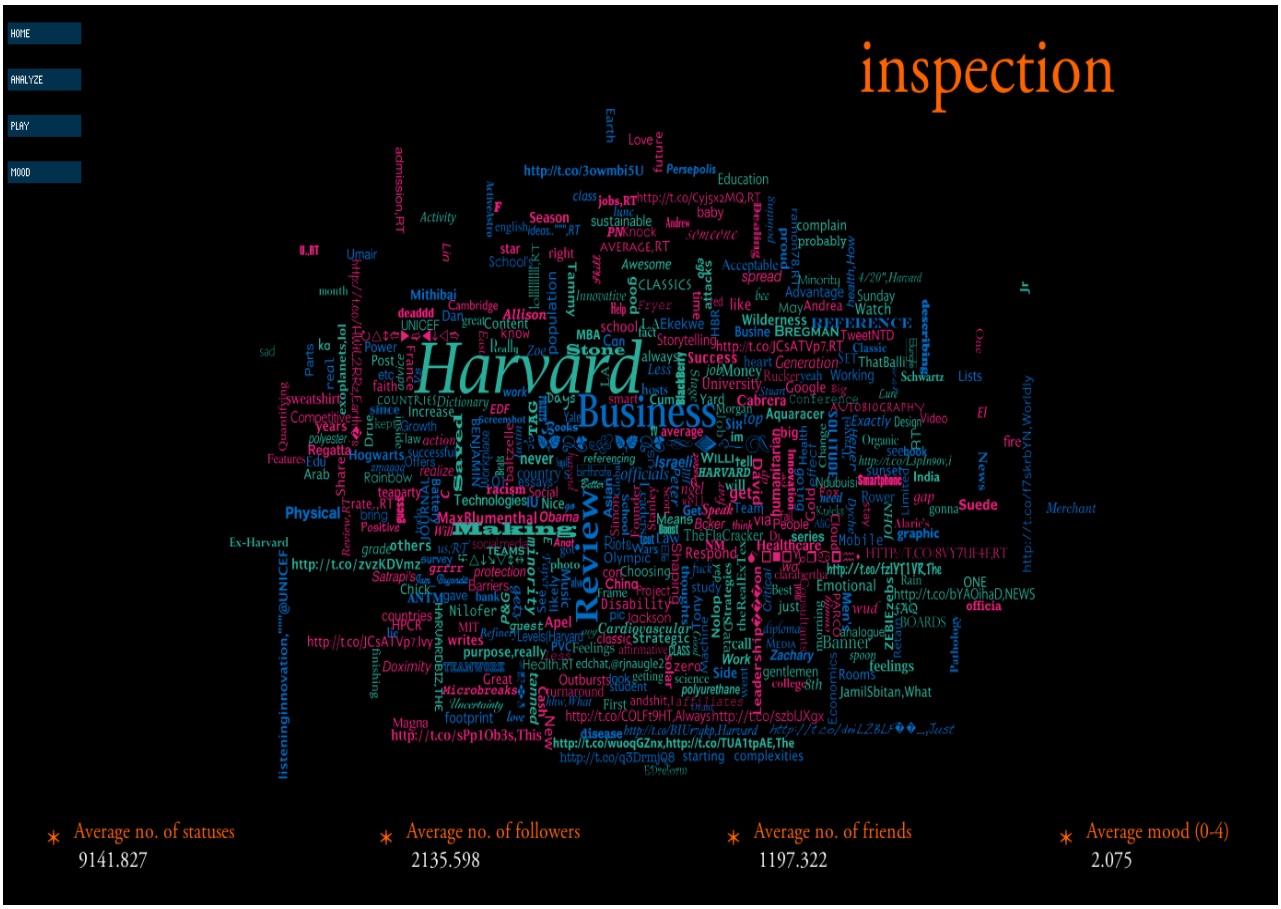


Figure 1: Tweetography, main view, complete with locations of all the tweets.

Figure 2: The user is allowed to click on certain tweets, for later data inspection.

Figure 3: This is the main panel, which allows for a multitude of features, both a time series generation and a mood analysis.

Figure 4: The map itself, a tiled view from TileMill, allows zoom rendering and is complete with national borders and major city names.


HOME
ANALYZE
PLAY
MOOD
inspection
Harvard
Business
Review
Making
Stone
Physical
Saved
Average no. of statuses
9141.827
Average no. of followers
2135.598
Average no. of friends
1197.322
Average mood (0-4)
2.075


April 20
3:10 (24 hr)