

Distilling Massive Amounts of Data into Simple Visualizations: Twitter Case Studies

Miguel Rios and Jimmy Lin

Twitter, Inc.

@miguelrios @lintool

Introduction

Twitter is a communications platform on which users can send short, 140-character messages, called “tweets”, to their “followers” via a number of mechanisms, including web clients, mobile clients, and SMS. As of March 2012, Twitter has over 140 million active users worldwide, who collectively post over 340 million tweets per day. Particularly salient is the real-time nature of these global conversations, which rapidly evolve to reflect breaking events such as major earthquakes (e.g., Japan, March 2011) and deaths of prominent figures (e.g., Steve Jobs, October 2011).

From this large user base we gather tens of terabytes of data per day, containing records of what users tweet (and when and where), which tweets they interact with, and a host of other activities. How do we derive insights from these massive amounts of data? This responsibility falls primarily to the data analytics group within the company. Organizationally, the group is divided into two logical, but tightly-coupled teams: one handles the infrastructure necessary to manage massive data warehouses, the other consists mostly of “data scientists”, a new breed of engineers charged with generating insights from the data.

Ultimately, the data scientist’s job is to analyze massive amounts of data, interpret “what the data say”, and distill the bits into *actionable* insights that steer the direction of the company: what web site elements to refine, what features to develop, what markets to pursue, etc. To accomplish this, data visualization is an indispensable tool. At a mundane level, the group provides dashboards to enable stakeholders to browse through large amounts of multi-dimensional data, including interactive “drill downs” and “roll ups”, overlaid with projections and other derived data. This is quite similar to cube materialization in online analytical processing (OLAP) tasks for business intelligence, and is supported by a number of commercial off-the-shelf packages (although for tight integration with our infrastructure, our system is entirely developed in-house).

Beyond simple dashboards, data scientists often build one-off visualizations that are the result of a specific task, usually a business question. For example, in 2010 when Twitter was outgrowing its datacenter footprint, we created a custom visualization that let engineers interactively examine server utilization statistics along with network bandwidth usage in order to better optimize resource allocation.

Finally, we occasionally produce visualizations intended for public consumption, which are presented on Twitter’s blog. These are less intended to generate insights for the business; rather, their primary purpose is to highlight the “pulse” of the global conversation on Twitter, often in reaction to major news events around the world.

All these visualizations, from simple line graphs to complex interactive browsing interfaces, share one common feature: although the ultimate product consists only of a few hundred to a few thousand data points, they are the distillation of gigabytes, and in some cases, terabytes of raw data. When viewing or interacting with a visualization, it is easy to forget all the “data crunching” that went into creating it. Giving the reader a sense of how this happens is the purpose of this paper. We begin with a brief overview of the analytics infrastructure that supports these tasks, and present three case studies describing how visualizations are created.

Twitter’s Analytics Stack

Twitter’s analytics infrastructure is built around Hadoop, the open-source implementation of MapReduce (Dean and Ghemawat 2004), which is a popular framework for large-scale distributed data processing. Our central data warehouse is built around a large Hadoop cluster. Data arrive in the Hadoop Distributed File System (HDFS) via a number of real-time and batch processes: bulk exports from frontend databases, application logs, and many other sources.

Although Hadoop is implemented in Java, analytics is largely performed using Pig, a high-level dataflow language that compiles into physical plans that are executed on Hadoop (Olston et al. 2008). Pig provides concise primitives for expressing common operations such as projection, selection, group, join, etc. This conciseness comes at low cost: Pig scripts approach the performance of programs directly written in Hadoop Java. Yet, the full expressiveness of Java is retained through a library of custom user-defined functions that expose core Twitter libraries (e.g., for extracting and manipulating parts of tweets).

To provide the reader with a flavor of what Pig scripts look like, here is a (simplified) script for counting the distribution of terms in tweets (say, all tweets in 2011):

```
A = load '/tweets/2011' using TweetLoader();  
B = foreach A generate
```

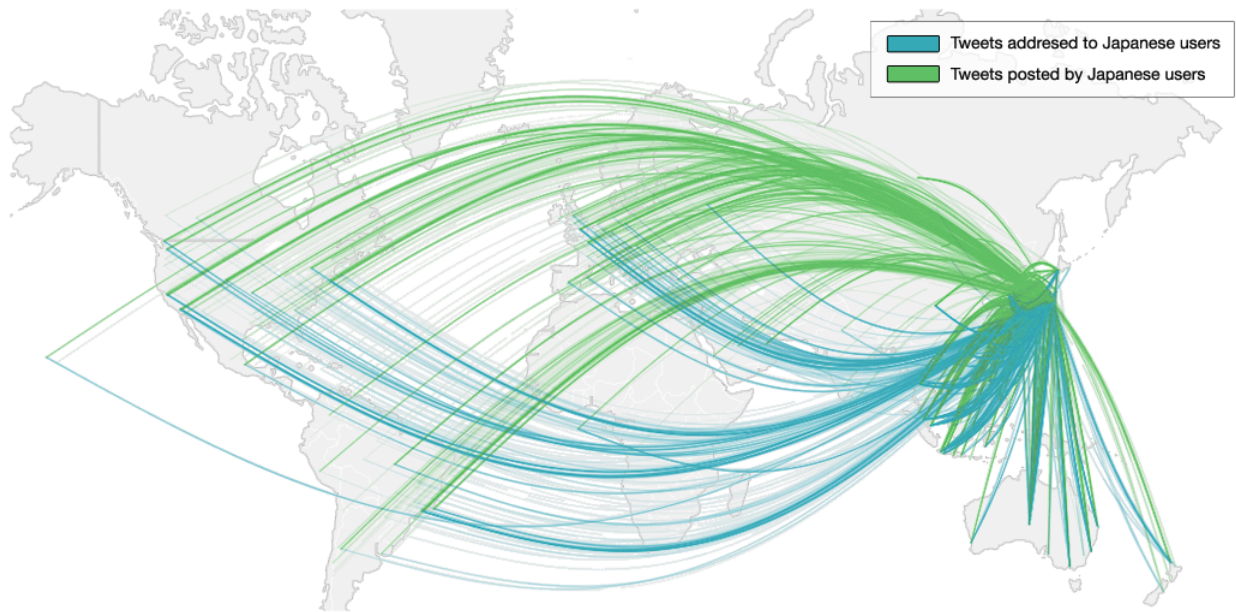



Figure 2: Tweet flow during the 2011 Tohoku Earthquake. Green lines represent tweets posted by Japanese users and blue lines represent tweets addressed to Japanese users. Together, these define the global lines of communication via which news is shared. See the animated version at http://bit.ly/jp_quake_viz

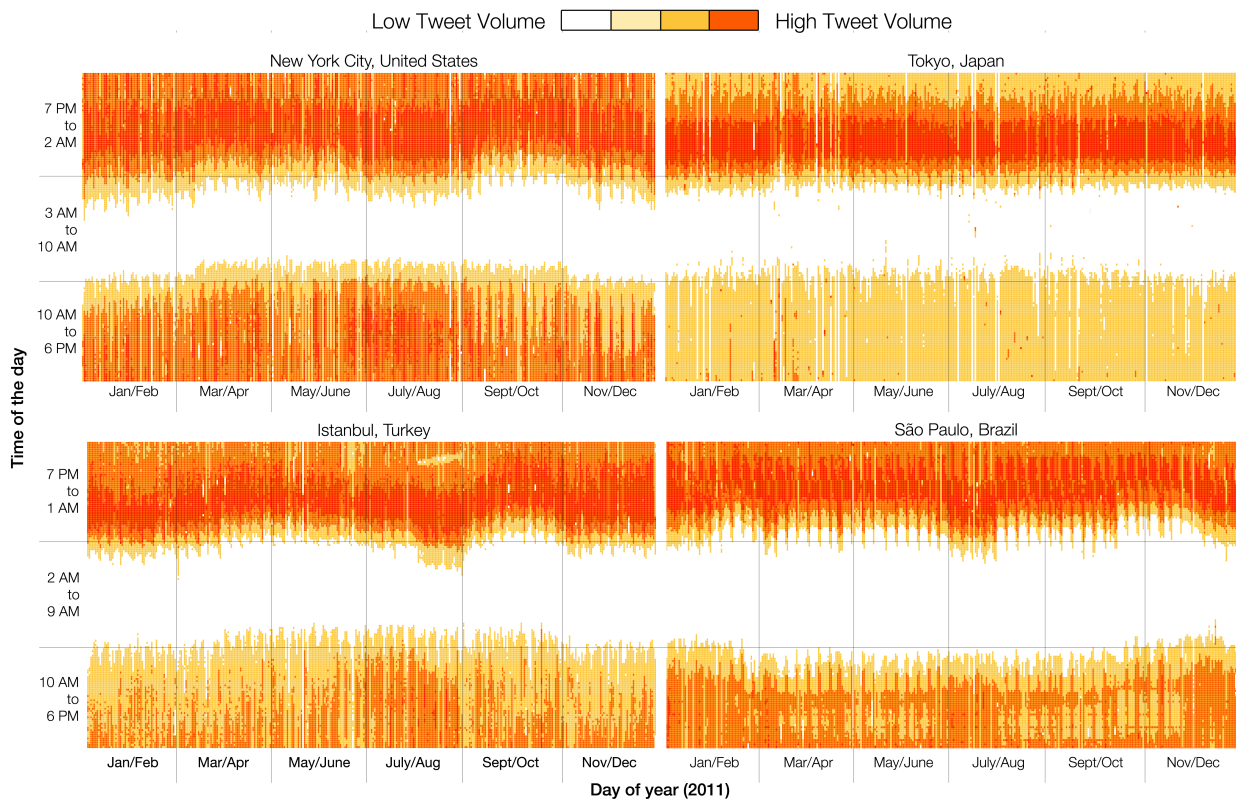


Figure 3: Seasonal variation of tweeting patterns for users in four different cities. The gradient from white to yellow to red indicates amount of activity (light to heavy). Each tile in the heatmap represents five minutes of a given day and colors are normalized by day.

Once the earthquake occurs, we see a 500% spike in traffic as users share the news with friends around the world.

Once again, raw data for creating this animation came from the output of Pig scripts. First, tweets were filtered to retain only the @-replies. These were then joined with user data, which contain locations (based on IP addresses). From this, we arrived at a distilled dataset with source and destination locations for the tweets. Joins are a primitive operation in Pig, and is as easy as specifying the left and right relations and the join attribute.

These intermediate data were then exported and processed using Javascript to create a browser-based visualization with `processing.js`.⁴ A static map was embedded in the background to help users understand the origins and destinations of the tweets. In the foreground, circles representing groups of tweets navigate from the locations where the tweets were posted to their destinations. In the animation, each circle leaves a trail behind it when it moves, and in aggregate, these trails define the global lines of communication.

Tweeting Patterns: Seasonal Variations

In the last case study, we show a visualization of daily tweeting patterns for 2011, focusing on four locations (New York City, Tokyo, Istanbul, and São Paulo). This was developed internally to understand why growth patterns in tweet-production experience seasonal variations. As usual, raw data came from Pig output. Tweets were filtered based on location and then bucketed by day and time into 5-minute intervals. Finally, counts for each bucket were generated, resulting in a raw dataset containing, day, time bucket, and the number of tweets that were published. We visualized the data using R, since the visualization did not require any animation or user interaction, and R's `ggplot2` has good support for heatmaps using `geom.tiles`.⁵

The final output is shown in Figure 3. We see different patterns of activity between the four cities. For example, waking/sleeping times are relatively constant throughout the year in Tokyo, but the other cities exhibit seasonal variations. We see that Japanese users' activities are concentrated in the evening, whereas in the other cities there is more usage during the day. In Istanbul, nights get shorter during August; São Paulo shows a time interval during the afternoon when tweet volume goes down, and also longer nights during the entire year compared to the other three cities.

These observations are not conclusive, but they suggest further lines of exploration: for example, why do these patterns differ? Is it a cultural effect? This might, in turn, trigger more analyses to help us better understand user behavior. Understanding these seasonal patterns help us predict future activity and also help us prepare for growth and strategic investments in international markets.

Visualization Meets Big Data

At a high level, our strategy for data visualization can be divided into three main steps: high-level planning, gathering raw data, and creating the visualization itself. The first

⁴ www.processingjs.org

⁵ had.co.nz/ggplot2/geom.tile.html

step depends on the creativity of the data scientist and the question at hand. The second step involves analyzing massive amounts of data—usually with Pig in the datacenter. Pig scripts are executed in batch mode, which means that results are not available immediately (job latency is dependent on data size and cluster load). In contrast, the third step of creating the visualization is usually performed on a laptop and thus benefits from a rapid development cycle (i.e., a browser refresh when working with web-based tools). This creates an impedance mismatch: waiting for Pig output, copying results out of the datacenter. Furthermore, this transfer often involves burdensome conversion between data formats (e.g., from delimited files to JSON and back).

Naturally, refining visualizations require repeated iterations between the second and third steps (e.g., we often don't know about data processing errors until we visualize the data). Thus, transitioning between datacenter- and local-processing is a source of friction: we identify this as a large gap in the state of the art and thus represents an opportunity for research. In short, how do we marry big data and visualization? This challenge applies to both the development cycle and particularly for interactive visualizations. The standard strategy is to pre-materialize all data that one could potentially imagine visualizing, but this places an upper bound on the amount of data possible (megabytes in browser-based toolkits with current technologies). Furthermore, what if the user requests data that has not been pre-materialized? Going back to Pig to generate more results will yield long latencies, creating a subpar user experience. This challenge cannot be solved by simply trying to improve the speed of backend data processing components, but requires genuine coordination between the local client and the remote data warehouse to hide the inevitable latencies.

A natural consequence of the above arguments, we believe, is that “big data” skills are now critical for both researchers and practitioners of data visualization. Data scientists need an eye for how to “tell stories” with data, but since data warehouses today are measured in petabytes, to be effective they must also be able to manipulate raw data themselves. This requires not only a background in human-computer interaction, some feel for distributed processing, but also competence in statistics and machine learning. The ideal data scientist is part engineer, part artist—but in our experience, an individual with this combination of skills is difficult to find, primarily because no existing curriculum trains students in all these areas adequately. Nevertheless, such individuals will become increasingly valuable to organizations that struggle with ever-growing amounts of data.

References

- Dean, J., and Ghemawat, S. 2004. MapReduce: Simplified data processing on large clusters. In *OSDI*.
- Dörk, M.; Gruen, D.; Williamson, C.; and Carpendale, S. 2010. A visual backchannel for large-scale events. *IEEE Transaction on Visualization and Computer Graphics* 16(6): 11291138.
- Olston, C.; Reed, B.; Srivastava, U.; Kumar, R.; and Tomkins, A. 2008. Pig Latin: A not-so-foreign language for data processing. In *SIGMOD*.