

Using the Twitter API

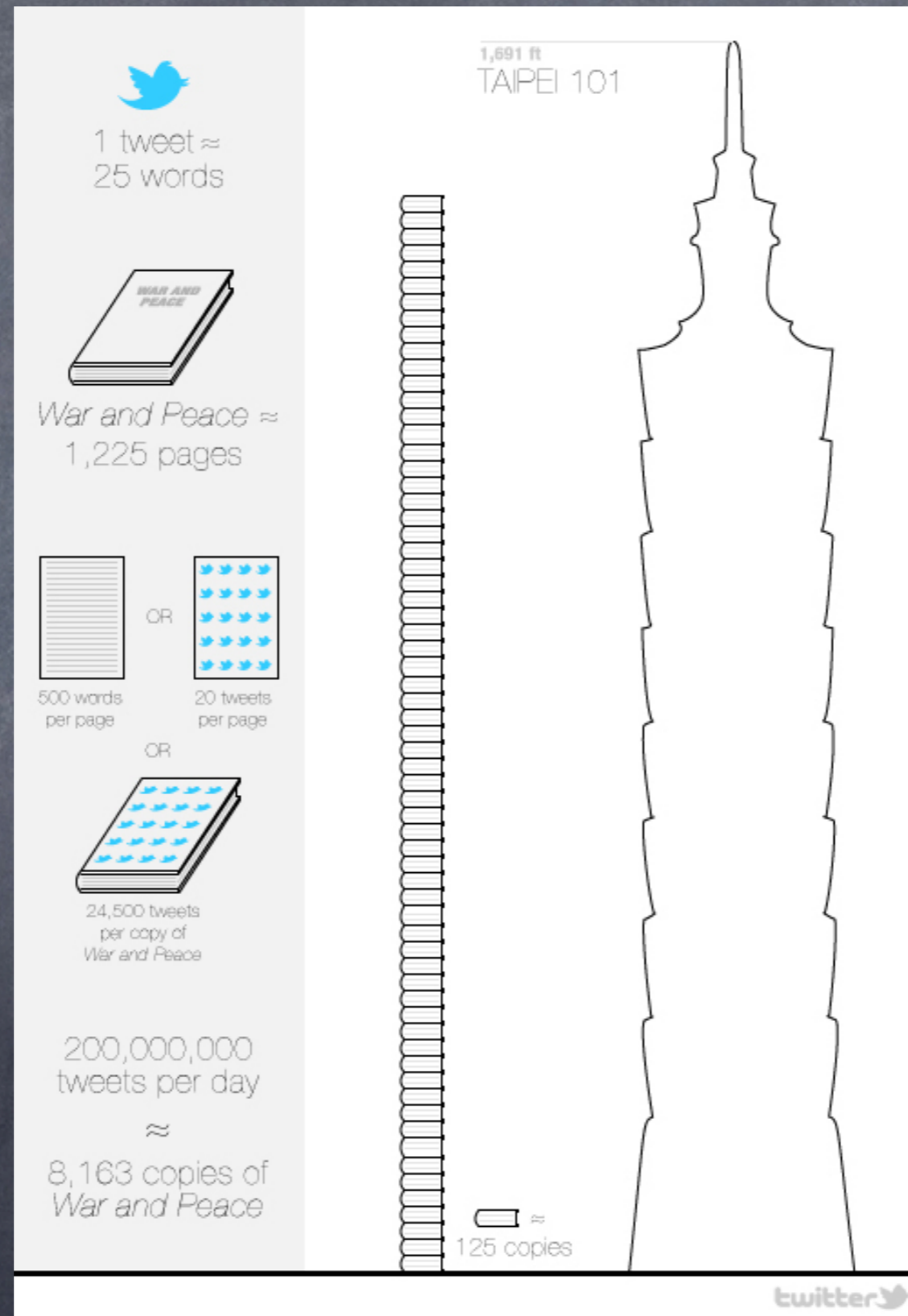
Rion Snow
@rion

Preparation for demos

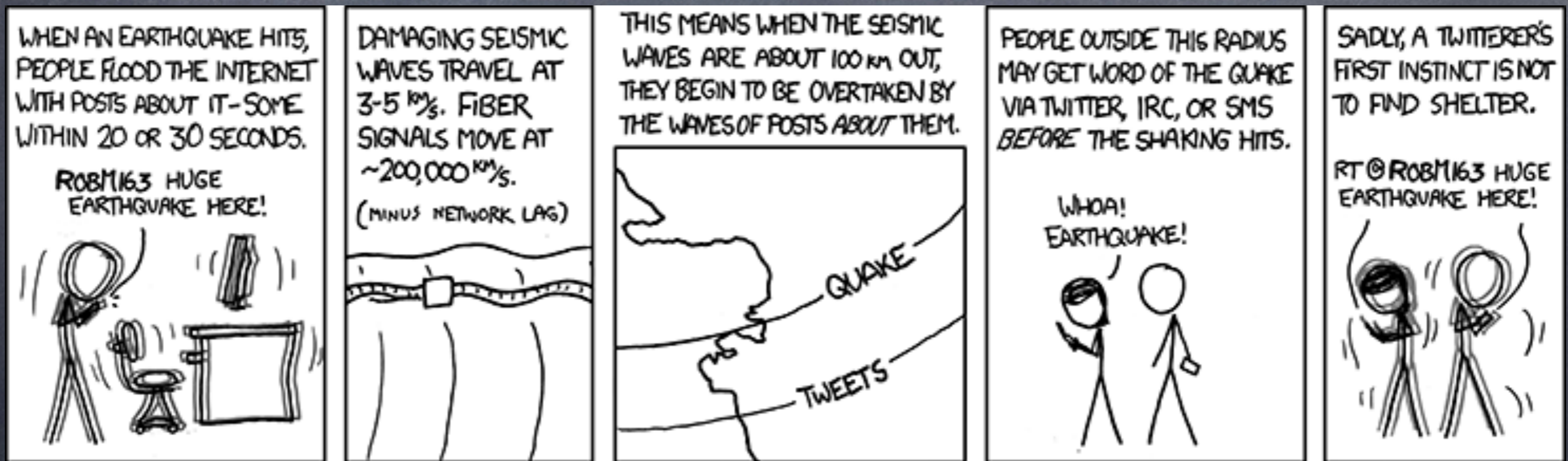
- 1) Download latest stable twitter4j library:
`curl http://twitter4j.org/en/twitter4j-2.2.6.zip > twitter4j-2.2.6.zip`
- 2) Unzip into a folder:
`unzip twitter4j-2.2.6.zip`
- 3) Follow instructions in `bin/readme.txt`
- 4) Install tweepy:
`easy_install tweepy`
- 5) follow instructions at:
<http://talkfast.org/2010/05/31/twitter-from-the-command-line-in-python-using-oauth>

Motivation

Not just big data...



It's also fresh data



<http://xkcd.com/723/>

Not just earthquakes...

 **@jkrums**
Janis Krums

http://
in t
pic

 **Sohaib Athar**
@ReallyVirtual

Helico
1AM (i

← Reply

3,428
RETWEETS

12:58 PM -

 **Occupy Oakland**
@OccupyOakland

#occupyoakland at
suprise assault. tea
shotguns, flash bar
injured.

← Reply ↻ Retweet ★ Fav

4,432 RETWEETS	111 FAVORITES	
--------------------------	-------------------------	---

5:20 AM - 25 Oct 11 · Embed this



 **Stefanie Gordon**
@Stefmara

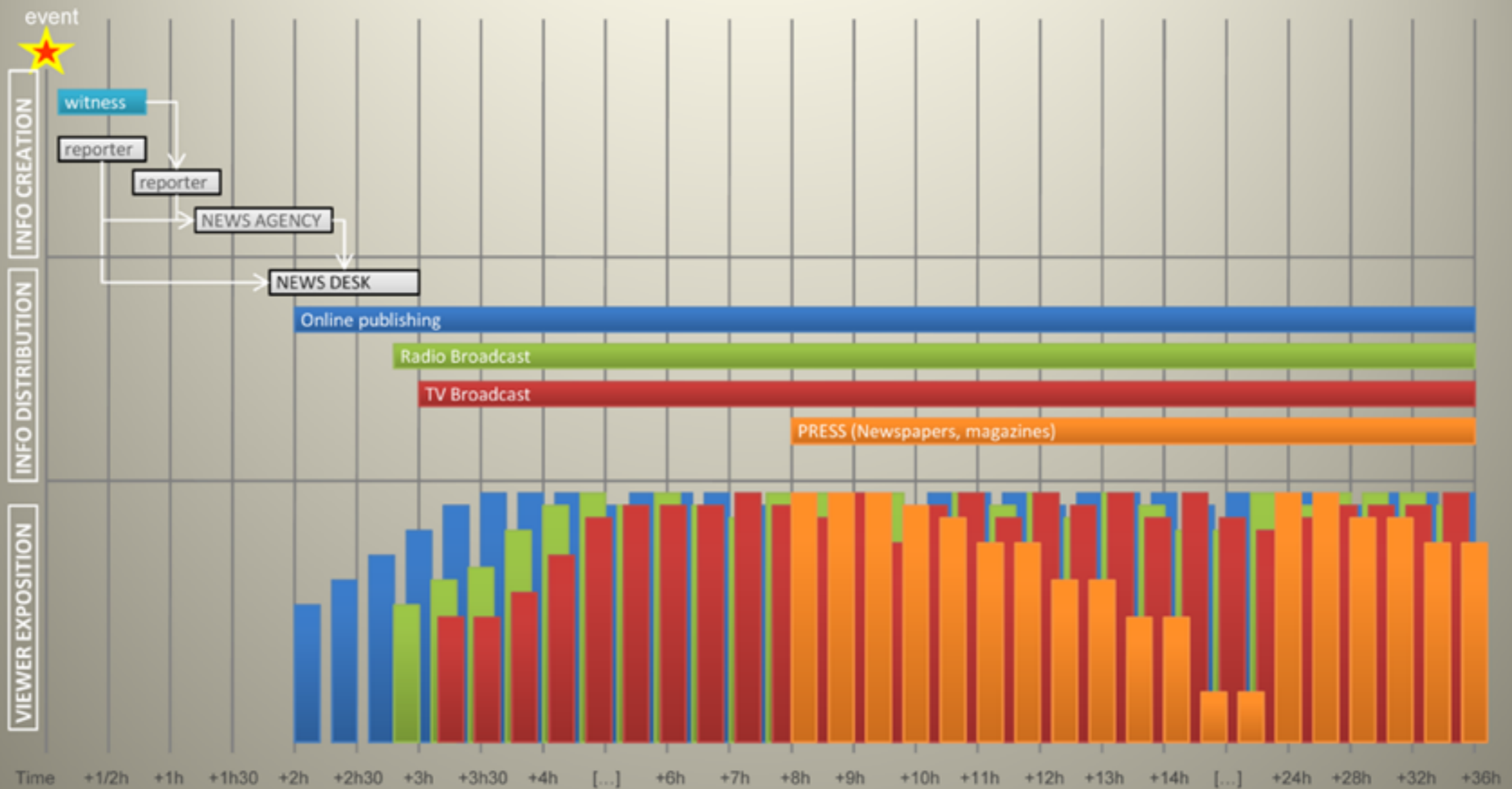
Here's another Photo of the shuttle from my plane. <http://twitpic.com/4yg6hs>

← Reply ↻ Retweet ★ Favorite



By Stefanie Gordon @Stefmara

Information creation & circulation, before Twitter



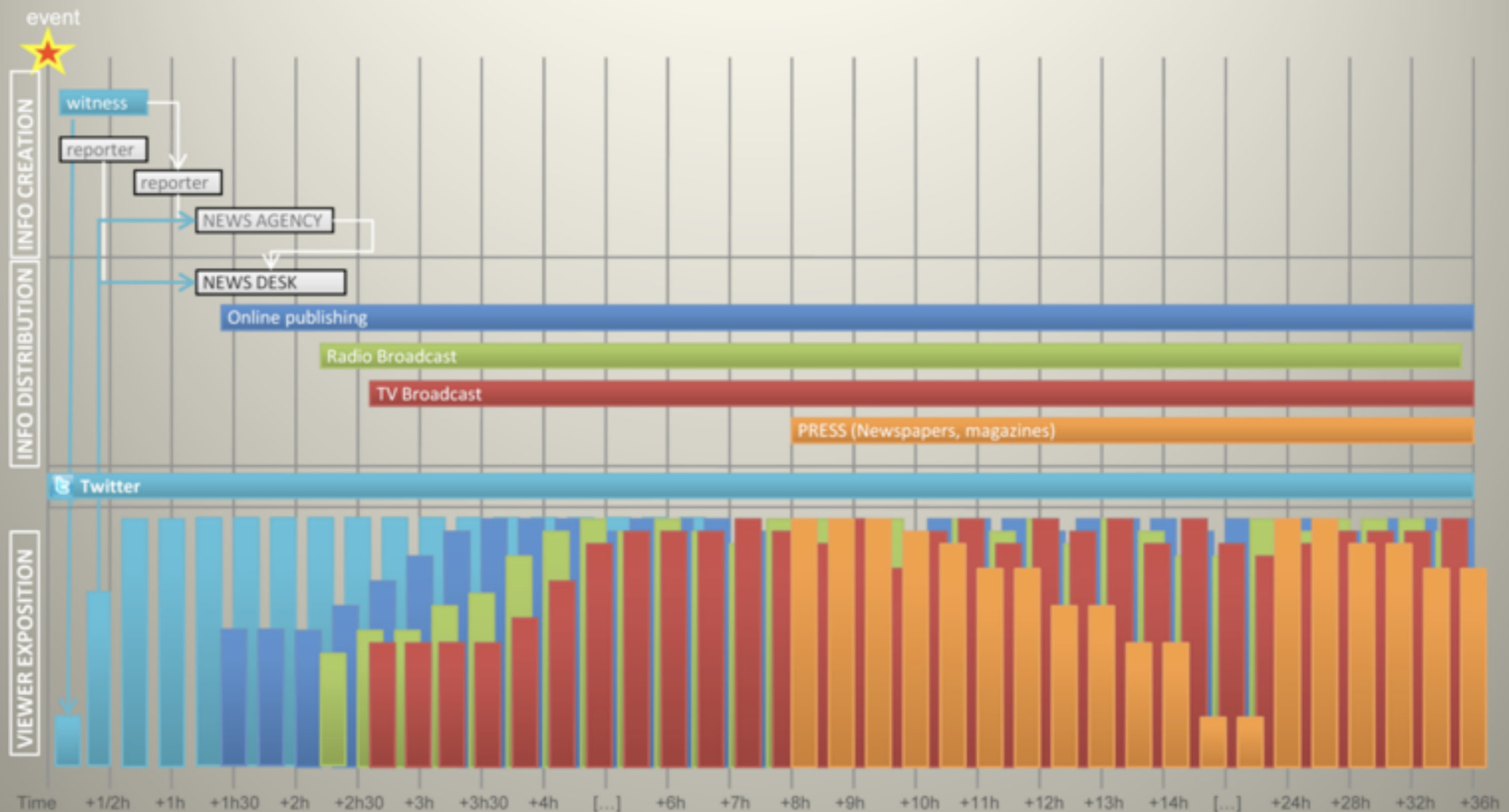
Notes :

- Online publishing usually faster than other media, in part due to automatic wire republication on websites.
- Press exposition comes late due to printing, and can not be updated before the following edition
- “Viewer exposition” volume guessed based on a typical day at work : Internet, then radio when commuting, then TV at home.



Burson-Marsteller

Information creation & circulation, after Twitter



Notes :

- Witnesses can immediately publish their story & pictures on Twitter
- Reporters, news agencies, news desks feed on Twitter to find new stories
- It may increase the time-to-air time between the event and its actual airing by the media



Burson-Marsteller

Motivation

Summary

- Anatomy of a Tweet and User
- The Twitter API: REST and Streaming
- Recommended libraries: Twitter4J & tweepy
- Applications built on the Streaming API

Anatomy of a tweet [I]



<https://dev.twitter.com/docs/platform-objects/tweets>

https://api.twitter.com/1/statuses/show.json?id=172070369035956224&include_entities=true

Anatomy of a tweet [II]



```
created_at: "Tue Feb 21 21:29:07 +0000 2012",
id: 172070369035956220,
id_str: "172070369035956224",
text: "The \"http://\" at the beginning of URLs is a command to the browser. It stands for \"head to this place:\" followed by two laser-gun noises.",
source: "web",
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: 63846421,
  geo: null,
  coordinates: null,
  place: null,
  contributors: null,
  retweet_count: 2128,
  entities: {
    hashtags: [ ],
    urls: [ ],
    user_mentions: [ ]
  },
}
```

Anatomy of a user [I]



The image shows a screenshot of a Twitter profile for the user 'twitterapi'. The profile is verified and has a blue checkmark. The bio reads: 'The Real Twitter API. I tweet about API changes, service issues and happily answer questions about Twitter and our API. Don't get an answer? It's on my website.' The website link is 'http://dev.twitter.com'. The user has 988,982 followers and is following 33 accounts. The profile picture is a blue gear with a white Twitter bird inside. There are also some small icons for 'GET /jobs' and a row of profile pictures for the accounts being followed.

twitterapi Twitter API ✓
The Real Twitter API. I tweet about API changes, service issues and happily answer questions about Twitter and our API. Don't get an answer? It's on my website.
<http://dev.twitter.com>

Followers **988,982** Following **33**

GET /jobs

<https://dev.twitter.com/docs/platform-objects/users>

https://api.twitter.com/1/users/show.json?screen_name=twitterapi&include_entities=true

Anatomy of a user [II]

```
id: 6253282,
id_str: "6253282",
name: "Twitter API",
screen_name: "twitterapi",
location: "San Francisco, CA",
url: "http://dev.twitter.com",
description: "The Real Twitter API. I tweet about API changes, service issues and happily answer questions about Twitter",
protected: false,
followers_count: 1217031,
friends_count: 31,
listed_count: 10784,
created_at: "Wed May 23 06:01:13 +0000 2007",
favourites_count: 25,
utc_offset: -28800,
time_zone: "Pacific Time (US & Canada)",
geo_enabled: true,
verified: true,
statuses_count: 3336,
lang: "en",
status: {
  created_at: "Thu Sep 06 17:55:54 +0000 2012",
  contributors_enabled: true,
  is_translator: false,
  profile_background_color: "CODEED",
  profile_background_image_url: "http://a0.twimg.com/images/themes/theme1/bg.png",
  profile_background_image_url_https: "https://si0.twimg.com/images/themes/theme1/bg.png",
  profile_background_tile: false,
  profile_image_url: "http://a0.twimg.com/profile_images/2284174872/7df3h38zabcvjlynfe3_normal.png",
  profile_image_url_https: "https://si0.twimg.com/profile_images/2284174872/7df3h38zabcvjlynfe3_normal.png",
  profile_banner_url: "https://si0.twimg.com/profile_banners/6253282/1347053495",
  profile_link_color: "0084B4",
  profile_sidebar_border_color: "CODEED",
  profile_sidebar_fill_color: "DDEEF6",
  profile_text_color: "333333",
  profile_use_background_image: true,
  show_all_inline_media: false,
  default_profile: true,
  default_profile_image: false,
  following: null,
  follow_request_sent: null,
  notifications: null
}
```



The screenshot shows the Twitter profile for 'twitterapi'. The profile picture is a blue gear with a white bird. The name is 'twitterapi' with a verified badge. The bio reads: 'The Real Twitter API. I tweet about API changes, service issues and happily answer questions about Twitter and our API. Don't get an answer? It's on my website.' The website link is 'http://dev.twitter.com'. There is a green 'Following' button. Below the bio, it shows 'Followers 988,982' and 'Following 33'. At the bottom, there are icons for 'GET /jobs' and a row of profile pictures of people being followed.

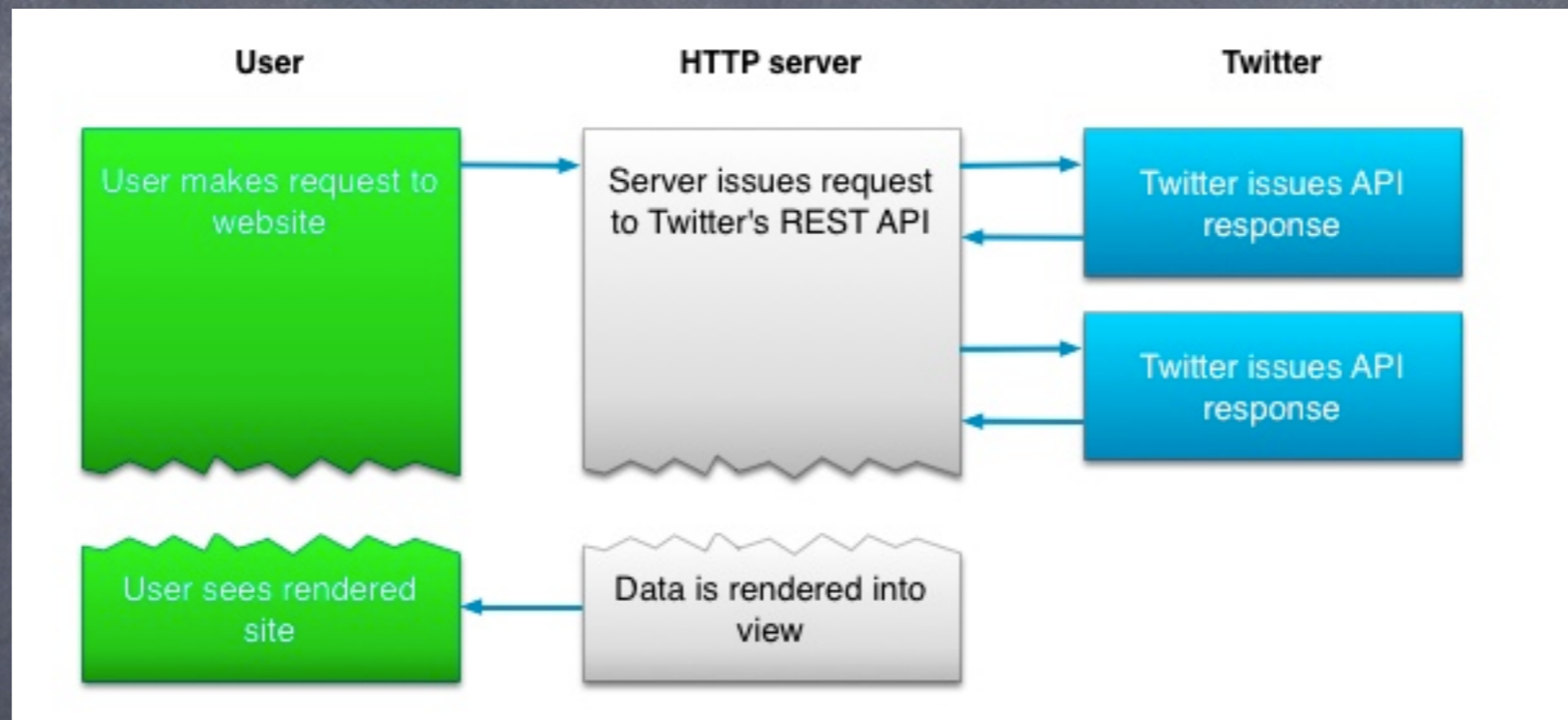
API

REST API

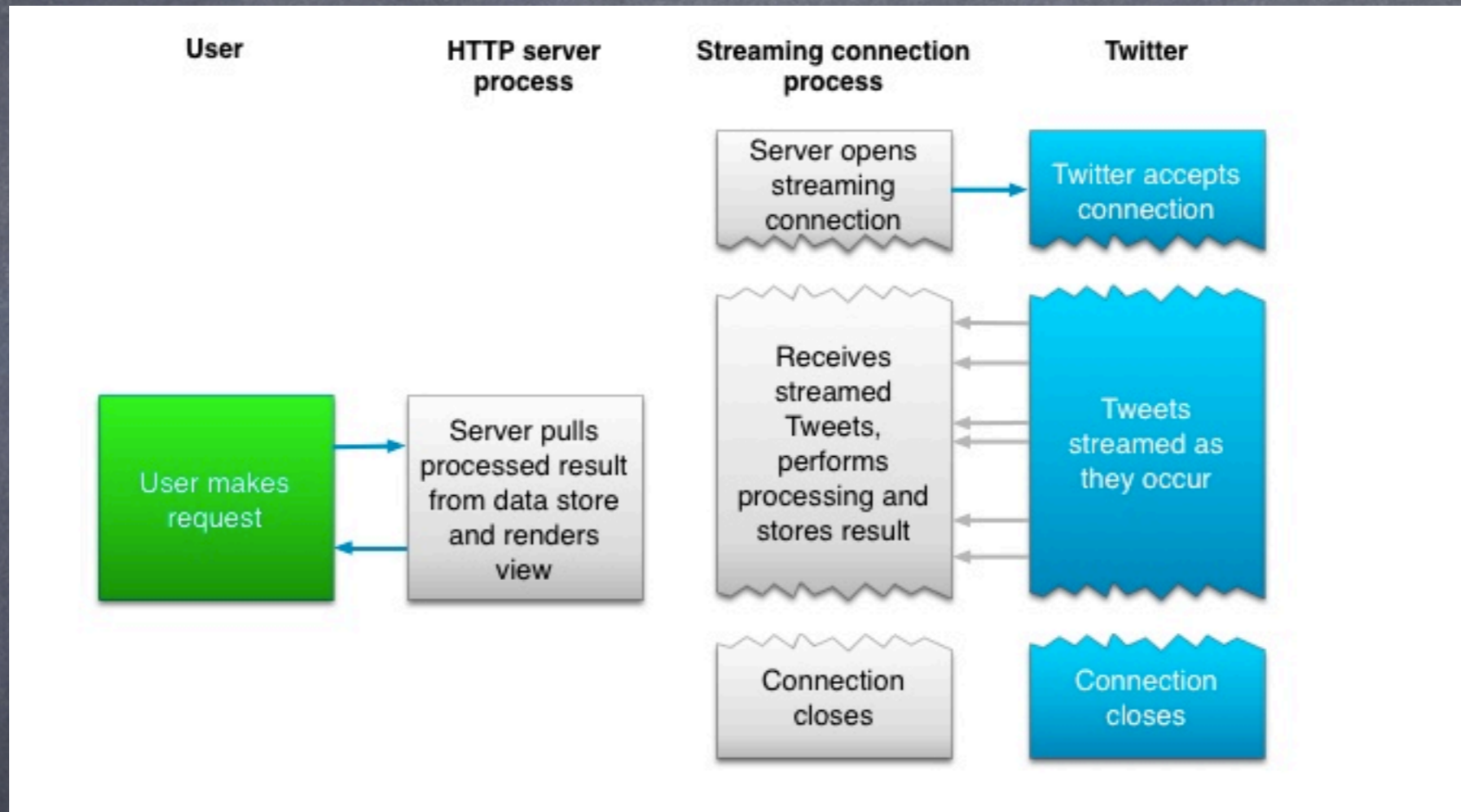
vs.

Streaming API

REST



Streaming



The REST API

- Getting info about a tweet: statuses/show
- Getting info about a user: users/show
- Getting a user's tweets: users/statuses
- Getting a user's timeline: users/timeline
- Getting a user's follows: users/follows

<https://dev.twitter.com/docs/api/1.1>

Demo: fetching a tweet

Accessing a Tweet: Twitter4j

```
Twitter twitter = new TwitterFactory().getInstance();
Status status = twitter.showStatus(Long.parseLong(args[0]));
System.out.println("@ " + status.getUser().getScreenName()
    + " - " + status.getText())
```

But also:

```
status.getRetweetCount(),
status.getGeoLocation(),
status.getInReplyToId(),
...
```

<http://twitter4j.org/en/javadoc/twitter4j/Status.html>

Accessing a Tweet: tweepy

```
username = "..."  
password = "..."  
basic_auth = tweepy.BasicAuthHandler(username, password)  
api = tweepy.API(basic_auth)  
status = api.get_status(172070369035956224)  
print '%s' % status.text
```

But also:

```
status.retweet_count,  
status.geo_location,  
status.in_reply_to_id  
...
```

<https://github.com/tweepy/tweepy/blob/master/tweepy/models.py>

Demo: fetching a user

Accessing a User: Twitter4J

```
Twitter twitter = new TwitterFactory().getInstance();
User user = twitter.showUser(args[0]);
if (user.getStatus() != null) {
    System.out.println("@" + user.getScreenName() + " - "
        + user.getStatus().getText());
}
```

But also:

```
user.getFollowersCount(),
user.getStatusesCount(),
user.getLocation(),
...
```

<http://twitter4j.org/en/javadoc/twitter4j/User.html>

Accessing a User: tweepy

```
username = "..."  
password = "..."  
basic_auth = tweepy.BasicAuthHandler(username,  
password)  
api = tweepy.API(basic_auth)  
user = api.get_user('twitterapi')  
print user.name, ': ', user.status.text
```

But also:

```
user.followers_count,  
user.statuses_count,  
user.location,  
...
```

<https://github.com/tweepy/tweepy/blob/master/tweepy/models.py>

Demo: Trends and Search

Accessing trends and search: Twitter4J

```
//trends
Twitter twitter = new TwitterFactory().getInstance();
Trends trends = twitter.getLocationTrends(Integer.parseInt(args[0]));
System.out.println("Showing location trends for woeid:" + args[0]);
System.out.println("As of : " + trends.getAsOf());
for (Trend trend : trends.getTrends()) {
    System.out.println(" " + trend.getName());
}
System.out.println("done.");

//search
QueryResult result = twitter.search(new Query(args[0]));
List<Tweet> tweets = result.getTweets();
for (Tweet tweet : tweets) {
    System.out.println("@ " + tweet.getFromUser() + " - " + tweet.getText());
}
```

Accessing trends and search: tweepy

```
username = "..."  
password = "..."  
basic_auth = tweepy.BasicAuthHandler(username, password)  
api = tweepy.API(basic_auth)  
locations = api.trends_available()  
trends = api.trends_location(2487956)  
search = api.search('Ice Cube')
```

REST API --> Twitter4J

- `statuses/show:id.json` --> `showStatus(statusId)`
- `users/show.json` --> `showUser(userId)`
- `statuses/user_timeline.json` --> `getUserTimeline()`
- `statuses/home_timeline.json` --> `getHomeTimeline()`
- `followers/ids.json` --> `getFollowersIds()`

<http://twitter4j.org/en/api-support.html>

REST API --> tweepy

- statuses/show:id.json --> get_status(id)
- users/show.json --> get_user(id)
- statuses/user_timeline.json --> user_timeline(id)
- statuses/home_timeline.json --> home_timeline(id)
- followers/ids.json --> followers(id)

<http://packages.python.org/tweepy/html/api.html>

Streaming API

- Sample all public statuses: `statuses/sample`
- Sample filtered statuses: `statuses/filter`

<https://dev.twitter.com/docs/streaming-apis>

Demo: sampling the Streaming API

Sampling the Streaming API: Twitter4J

```
TwitterStream twitterStream = new TwitterStreamFactory().getInstance();
StatusListener listener = new StatusListener() {
    public void onStatus(Status status) {
        System.out.println("@ " + status.getUser().getScreenName() + " - " +
            status.getText());
    }...
};
twitterStream.addListener(listener);
twitterStream.sample();
```

Add per-status business logic to the "onStatus" function; extract tweet text, increment counters, etc.

Sampling the Streaming API: tweepy

```
import tweepy

auth1 = tweepy.auth.OAuthHandler('CONSUMER KEY', 'CONSUMER SECRET')
auth1.set_access_token('ACCESS TOKEN', 'ACCESS TOKEN SECRET')
api = tweepy.API(auth1)

class StreamListener(tweepy.StreamListener):
    def on_status(self, status):
        try:
            print '\n %s %s %s via %s\n' % (status.text, status.author.screen_name,
status.created_at, status.source)
        except Exception, e:
            # Catch any unicode errors while printing to console
            # and just ignore them to avoid breaking application.
            pass
```

Add per-status business logic to the "on_status" function; extract tweet text, increment counters, etc.

<http://andrewbrobinson.com/2011/07/15/using-tweepy-to-access-the-twitter-stream/>

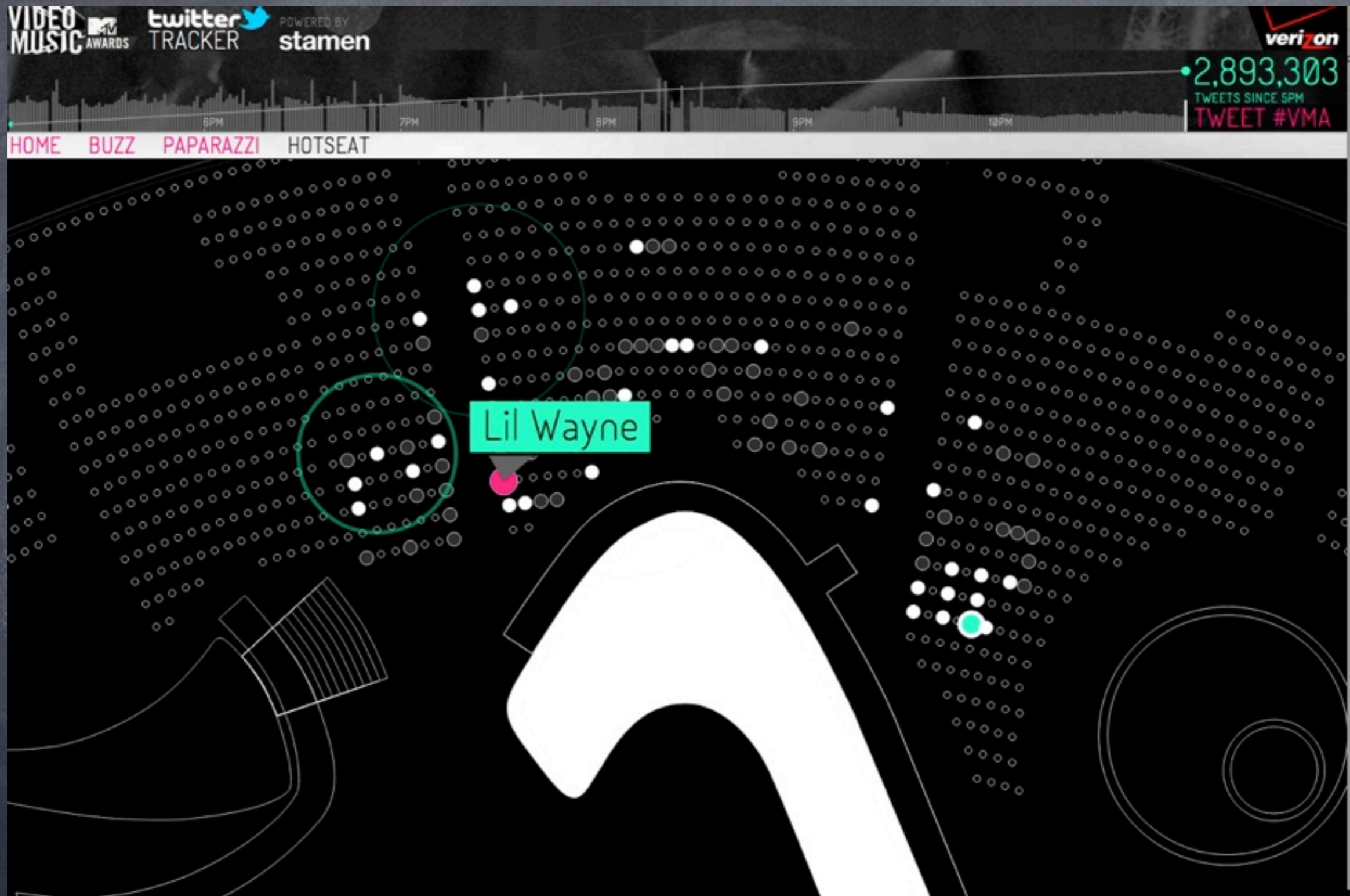
Libraries for accessing the Twitter API

- <https://dev.twitter.com/docs/twitter-libraries>

Applications and Visualizations

- VMA Twitter Tracker
- Twitter Political Index
- World of Tweets
- Earthquake Visualizations

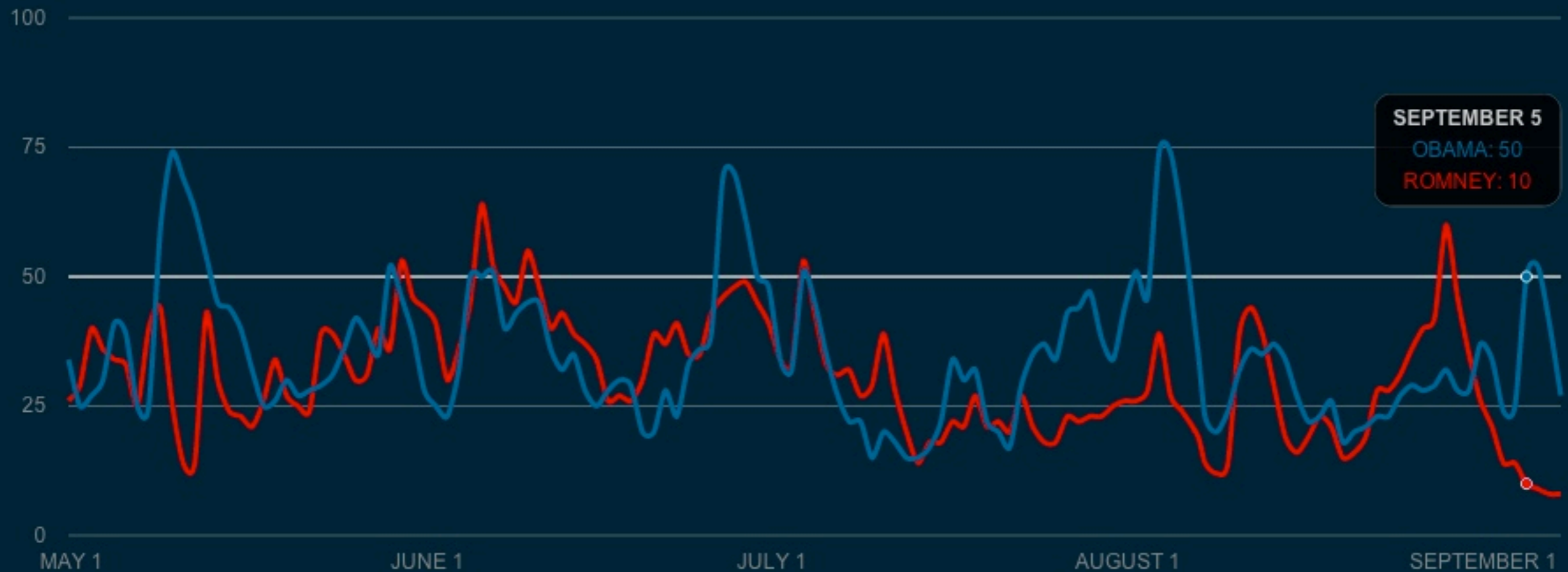
VMA Twitter Tracker



<http://vma-twittertracker.mtv.com/live/#hotseat>

Twitter Political Index

HISTORICAL INDEX



<https://election.twitter.com/>



• <http://aworldoftweets.frogdesign.com/>

2010 VMA live traffic

<http://vimeo.com/11302556>

March 2011 Japan Earthquake

<http://www.flickr.com/photos/twitteroffice/5885172082/>

More Resources

<https://dev.twitter.com/>

<https://dev.twitter.com/opensource>

<http://engineering.twitter.com/>

Characterizing Microblogs with Topic Models

Daniel Ramage

Stanford University
353 Serra Mall, Stanford, CA
dramage@cs.stanford.edu

Susan Dumais

Microsoft Research
One Microsoft Way, Redmond, WA
sdumais@microsoft.com

Dan Liebling

Microsoft Research
One Microsoft Way, Redmond, WA
danl@microsoft.com



Bill Gates Thank you to everyone at Cal! Excellent conversations and meetings with students. On to Stanford...
2:37 PM April 19

Characterizing Microblogs with Topic Models

Daniel Ramage, Susan Dumais, Dan Liebling

Abstract
As microblogging grows in popularity, services like Twitter are coming to support information gathering needs above and beyond their traditional role as social networks. But most users' interaction with Twitter is still primarily focused on their social graphs, forming the often inappropriate context of "people I follow" with "stuff I want to read." We characterize some information needs that the current Twitter interface fails to support, and argue for better representations of content for solving these challenges. We present a scalable implementation of a partially supervised learning model (Labeled LDA) that maps the content of the Twitter feed into dimensions. These dimensions correspond roughly to substance, style, status, and social characteristics of posts. We characterize users and tweets using this model, and present results on two information consumption oriented tasks.

Introduction
Millions of people turn to microblogging services like Twitter to gather real-time news or opinion about people, things, or events of interest. Such services are used for social networking, e.g. to stay in touch with friends and colleagues. In addition, microblogging sites are used as publishing platforms to create and consume content from sets of users with overlapping and disparate interests. Consider a hypothetical user @jane who follows user @frank because of the latter's posts about college football. However, @frank additionally uses Twitter to coordinate social arrangements with friends and occasionally posts political viewpoints. Currently, @jane has few tools to filter non-football content from @frank. In short, Twitter assumes that all posts from the people @jane follows are posts she wants to read. Similarly, @jane has a limited set of options for identifying new people to follow. She can look at lists of users in the social graph (e.g. those followed by @frank), or she can search by keyword and then browse the returned tweets' posters. However, it remains difficult to find people who are like @frank in general or - more challengingly - like @frank but with less social chatter or different political views.

Copyright © 2010, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

