Celebrities do not only make news –

they also spread them

Roy Reshef

How a famous film critic from Chicago, film and TV actors shape up the political mood in the US in the post-hurricane-Sandy pre-presedential-elections days. Analysis of realtime news spread on Twitter using the Retweet feature.

Preface

The emerging of social media has irreversibly changed the mechanism of news spread, both globally and nationally. If in the 1990's and the early 2000's the world would get glued to CNN for realtime breaking news feed, now it turns to Twitter. Storms, earthquakes, political events, pop concerts - social media has hundreds of millions of reporters, armed with smartphones, and the diffusion of news is as easy as one button click.

Definitions

This paper focuses on news spread via Twitter.
Assume userA posts a tweet, and userB finds it worthwhile to spread around. userB has the following options:

- Twitter's native retweet feature - userB reposts the original status, which is visible to userB's followers like this:
  userA original tweet text
  *retweeted by **userB***
- Quote. This is the old-school retweet mechanism, still supported by many Twitter client applications. The new tweet looks like:
  userB RT @userA: original tweet text
- Modified quote:
  Sometimes userB wishes to change the text.
  In some cases this is due to technical limitations, e.g. in case the inclusion of "RT @userA" brings the tweet length over the 140 character limit. In those cases the new tweet will look like:
  userB RT @userA: shortened tweet text
  In other cases, especially when userA has linked to a URL in the tweet, userB often wants to refer to that URL but include their own opinion or comment about it. In those cases the hat-tip or via convention is often used:
  userB new tweet text h/t @userA
  or
  userB new tweet text via @userA
- Reply:
  In some cases userB uses Twitter's native reply-to feature, but their reply includes enough information (e.g. a URL) from the original tweet to be considered a repost and not just a reply. Such a tweet would look like:
  userB @userA new tweet text

(visible by default only to userA)
or:
userB .@userA new tweet text
(visible by default to all of userB's followers)

- Plagiarism:
  And, of course, userB can simply copy userA's tweet without giving them any credit.

This paper focuses exclusively on the native retweet feature. This one can be determined unambiguously using Twitter API. All other mechanisms require much more comprehensive data collection, and with the exception of exact quote they also require some NLP (Natural Language Processing) to determine if a tweet is a repost of another tweet or not.

Note: all timestamps in this paper are in US/Eastern timezone.

Metrics and Goals

1. Time-to-masses: examine spread of news via the native retweet feature over time.
   How long did it take for a tweet to be retweeted by so many people? And to be potentially exposed to so many people?
2. Influential twitter users: identify people whose retweets contribute the most to the spread of a tweet.
3. Shortest path (in terms of hops and time) for a tweet to reach someone.

Related Work

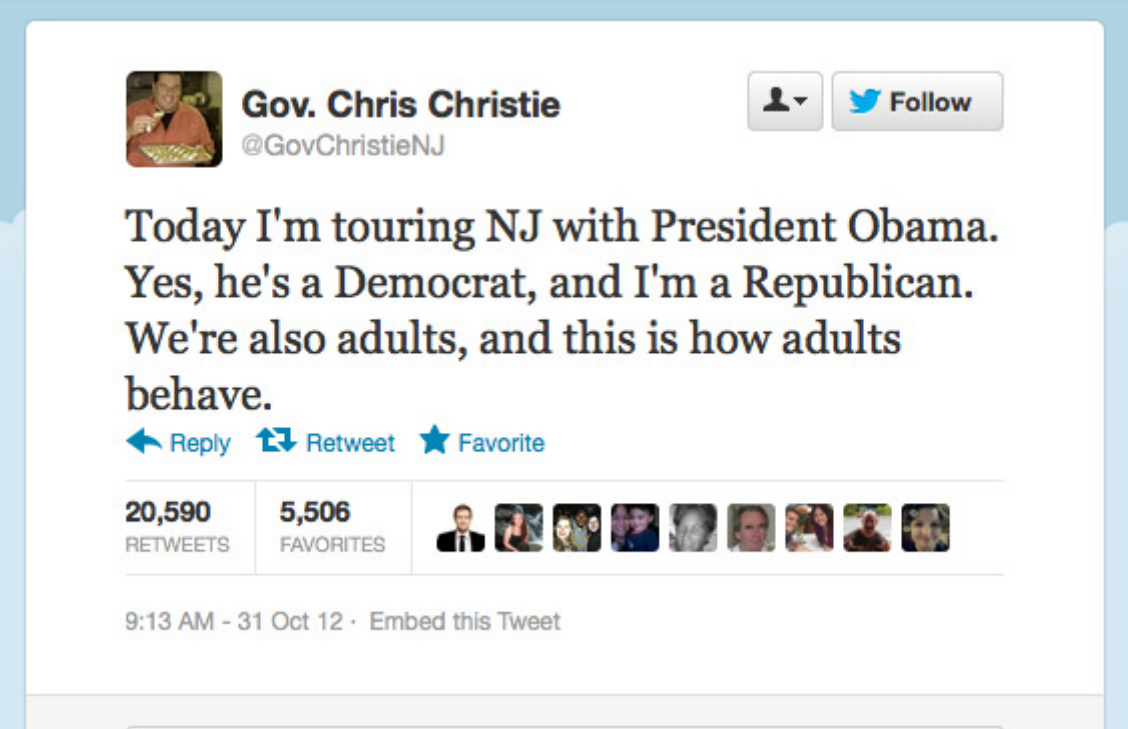
To the best of my knowledge, there was no attempt to measure the spread of news via Twitter using the metrics defined above.
[1] is trying to analyze the popularity of an article (a URL) linked in tweets, and based on characteristics of these articles to predict which articles will be popular on Twitter.
[2] is examining popularity of movies URLs and retweets about movies in the “critical weeks” of pre-release and release to predict success of movies.

Case Study

Following the landfall of hurricane Sandy on US East Coast on Monday, October 29, 2012, and in the days preceding the presidential elections (Tuesday, November 8), New Jersey Governor Chris Christie played an important role in US politics. Christie, a republican, praised the way campaigning President Obama, a democrat, has handled the devastating storm.

Governor Christie has his own verified Twitter account – https://twitter.com/GovChristie. Like with other celebrities, an unidentified impostor has created a fake account, assuming the Governor's identity (https://twitter.com/GovChristieNJ). This is not unheard of in Twittersphere, that's why Twitter came up with the "verified account" feature (an account verified by Twitter to be owned by the person it claims to be).

On Wednesday, October 31, President Obama and Governor Christie toured New Jersey to assess the damage caused by the hurricane. In the morning of that day, the "fake" Gov. Christie tweeted (https://twitter.com/GovChristieNJ/status/263629680731635712):



Twitter website, Monday, Nov. 5, 4:33 AM Eastern: 20,590 retweets.

https://github.com/tsipo/twitternews/blob/master/analysis/twitter-2012-11-05-4.33.31.png

This fake tweet has gone viral. It took a day or two to start, but within 5 days it has reached a remarkable tally of more than 20,000 retweets! It was also quoted on other social media and news aggregators (Facebook, Reddit etc.) – I have seen it myself first on Facebook. That triggered me to investigate how it spread viral.

Data Collection

1. Collect all native retweets of the abovementioned tweet in a time span of almost 5 days – from its original posting (Oct. 31, 9:13 AM) until Monday, Nov. 5, 4:15 AM. For each retweet, collect the retweeting user and the timestamp.
2. For each retweeter, collect basic information such as their number of followers and number of friends (people that they follow).
3. For each retweeter, collect the people they follow out of the group of the retweeters themselves.

Analysis

1. Draw a graph of accumulating number of retweets against timestamp.
2. Draw a graph of accumulating number of exposed twitter users (followers of people who retweet) against timestamp. This is a supremum – as it uses total number of followers (if user A follows users B and C who both retweet the same tweet, s/he will be counted twice).
3. Create a directed graph with weighted edges as follows:
    - Nodes are users.

- The weight of the edge i→j is **always** retweetTime(j) - retweetTime(i) (should always be positive).
- Let's denote the user whose tweet is in question (@GovChristieNJ) as ORIGIN.
- We add a node of UNKNOWN user with retweetTime identical to that of ORIGIN.
- We create a weighted edge (with weight 0.0) from ORIGIN to UNKNOWN.
- For each user j, we create a weighted edge from the (assumed) user i whose retweet "triggered" the retweet by user j: i→j. We determine the triggering user i as follows:
    i. We examine the timestamp of retweets.
    ii. If user j follows no other user (either ORIGIN or a retweeting user), we add a weighted edge from UNKNOWN to j. This could be the case that the user saw the tweet in Trending, in Twitter search or another network / news aggregator outside of Twitter.
    iii. Else, we filter all users who user j is following and ignore all of those who retweeted **after** user j has retweeted. If no users are left (who retweeted **before** user j), we add a weighted edge from UNKNOWN to j (obviously user j cannot follow ORIGIN in this case, as ORIGIN's timestamp is the earliest).
    iv. If some users are left after filtering, we assume that **the retweet which triggered user j's retweet is the one closest to user j's timestamp** (as it usually reappears on their timeline on Twitter website and Twitter client applications, even if other users they follow have retweeted it before). So we choose user i such that retweetTime(j) - retweetTime(i) is minimal (and positive).

4. Analyze the graph as follows:
    - Influential users: in our graph each user has indegree of 1 (that's how we created it), but the users with the highest outdegree are the most influential. I expect UNKNOWN to have a high outdegree (as some users came from other sources or due to missing data), but will it be beaten by a real user? Who are the most influential retweeters?
    - Shortest path:
      The shortest path of interest is the unweighted one – number of hops. How many hops did the tweet go (number of retweets) until it got to any retweeter. What is the diameter (longest path in the graph) – this can teach us on the length of a news-spread-chain.
      The weighted shortest path is simply the time elapsed since the original tweet until the retweet. It is also of interest, though it is influenced by the cutoff time of retweet collection (almost 5 days after the original tweet)
5. Export the graph to GML for loading into Gephi and further analysis.

Technical Implementation

Being a Java developer, I have implemented the above requirement in Java. The source code is available at https://github.com/tsipo/twitternews.
Data collection was done by using Twitter API and Topsy Otter API. For Twitter API I used Twitter4J (http://twitter4j.org), the leading Twitter Java client (I had the honour to contribute to it in 2010). I have used both version 2.2.6 and 3.0.0-SNAPSHOT of Twitter4J. The reason for that was that version 2.2.6 uses the deprecated Twitter API v1, and 3.0.0-SNAPSHOT uses Twitter API v1.1. In v1.1 Twitter has deprecated some calls I wanted to try out, and also changed the rate limit in a way that would make my run very long. For that reason the final version runs with Twitter4J 2.2.6 (Twitter API v1).

Topsy (http://topsy.com), a Twitter and other social media aggregator, has its Otter API. They have a python client library for it, but not a Java one. I have built the Java classes required to call the Otter API, with the help of Codehaus' Jackson JSON processor (http://jackson.codehaus.org/).
The graph generation and analysis was done with JGraphT (http://jgrapht.org/, https://github.com/jgrapht/jgrapht).

Data Collection Results

Retweets

Collecting retweets with Twitter API is not sufficient.
The API call (https://dev.twitter.com/docs/api/1.1/get/statuses/retweets/%3Aid, and also its v1 predecessor) returns up to 100 of the **latest retweets** (for the tweet in question, it returned 84). This is useless for our analysis, as we need all retweets.
Also the v1 calls https://dev.twitter.com/docs/api/1/get/statuses/%3Aid/retweeted_by, https://dev.twitter.com/docs/api/1/get/statuses/%3Aid/retweeted_by/ids (both deprecated in v1.1), which are paginated, returned at the time in question only the **latest** 701 retweets in total.

For this reason, I used Topsy Otter API. The Otter API requires a timeframe, and if there are too many retweets it returns total of 500. For this reason I took the "telescopic approach" – I tried a timeframe of 1 hour, and if it had 500+ retweets, I cut the timeframe by half and retried – and so on. In the busiest hours I had to cut the timeframe down to 7:30 minutes. So with timeframes of varying length, I covered the entire period Oct. 31, 9:13 AM – Nov. 5, 4:15 AM.

At the time my collection ended, Twitter showed 20,590 retweets (see screenshot above, https://github.com/tsipo/twitternews/blob/master/analysis/twitter-2012-11-05-4.33.31.png).

Topsy reported 17K retweets (http://topsy.com/twitter.com/GovChristieNJ/status/263629680731635712), probably as their collection and indexing services are a bit behind:

Topsy website, Monday, Nov. 5, 4:35 AM Eastern: 17K retweets.
https://github.com/tsipo/twitternews/blob/master/analysis/topsy-2012-11-05-4.35.12.png

Retweet issues

The collection of retweets using the "telescopic approach" yielded 9,303 tweets. This can be for several reasons.
I queried Topsy specifically for **native** retweets. It could be (I found no documentation of that) that the rewteet count of both Twitter and Topsy includes both retweets and quotes.
There was an "ominous" gap in retweets between the first retweet (Oct 31, 9:40 – about half an hour after the original tweet) and the next one – Nov. 1, 8:34 pm. This can be a result of missing tweets in either Topsy's collection (Topsy is connected to the Twitter firehose) and/or its indexing service. Could that tweet be ignored for so long, almost 36 hours? That could influence the results considerably.

Users Info

Next I turned to Twitter API to collect more information about the users. Namely, their Twitter user id, number of followers and friends. This was done by calling v1 https://dev.twitter.com/docs/api/1/get/users/lookup (or v1.1 https://dev.twitter.com/docs/api/1.1/get/users/lookup) and providing the list of user screen name which Otter API has yielded in the retweets. As this call can be done in bulks of 100 users at a time, only 94 calls (plus some retries due to errors) were needed – no rate limit issues here. As these calls were done during elections day (Nov. 8), it was subject to "Twitter is over capacity" HTML error pages, sometimes up to 20%-30% of the requests.

User issues

Twitter API returned info about 9271 of the 9303 users – the overall 32 users are probably suspended by Twitter.

Users Friends

Next came the call to determine who the retweeters follow (and find these in the retweeters group). The v1.1 API call https://dev.twitter.com/docs/api/1.1/get/friends/ids is rate-limited to 15 calls/15 minutes (60/hour) – way too slow. Therefore I reverted to the v1 https://dev.twitter.com/docs/api/1/get/friends/ids call, which shares the total rate limit pool of 350/hour. I ran no other calls and ran the application twice with different access tokens to speed things up.

User Friends issues

In total, 24 users of the 9271 are protected users – one cannot retrieve their friends. For these a "not authorized" error has been returned, in which case we need to skip collecting friends ids, of course (in case of timeout exceptions we simply retry). These users will have an incoming edge from UNKNOWN.

Data Collection Limitations

- Only about half of the documented retweets were returned by Topsy. If those missing retweets were missed by Topsy, the whole picture can change.
- Collection of data was done over a period of about 60 hours - first retweets, then users, then friendships (mainly due to the rate limit on the friendship collection). As data collection is a snapshot of the situation when it is done, things could change – i.e. friendships added/removed, users get suspended etc. – which impact the flow in the graph.

- The need to rely on deprecated Twitter API v1 just to avoid rate limit issues (can be regulated with Twitter but I had no time to arrange this).

Analysis

All analysis results are available at https://github.com/tsipo/twitternews/tree/master/analysis.

Analysis – retweets and followers over time

The first analysis was to generate a CSV file of **accumulating retweets** and **accumulating potential readers** (supremum) over time. This file was imported into an Excel sheet, https://github.com/tsipo/twitternews/blob/master/analysis/totals.xlsx.

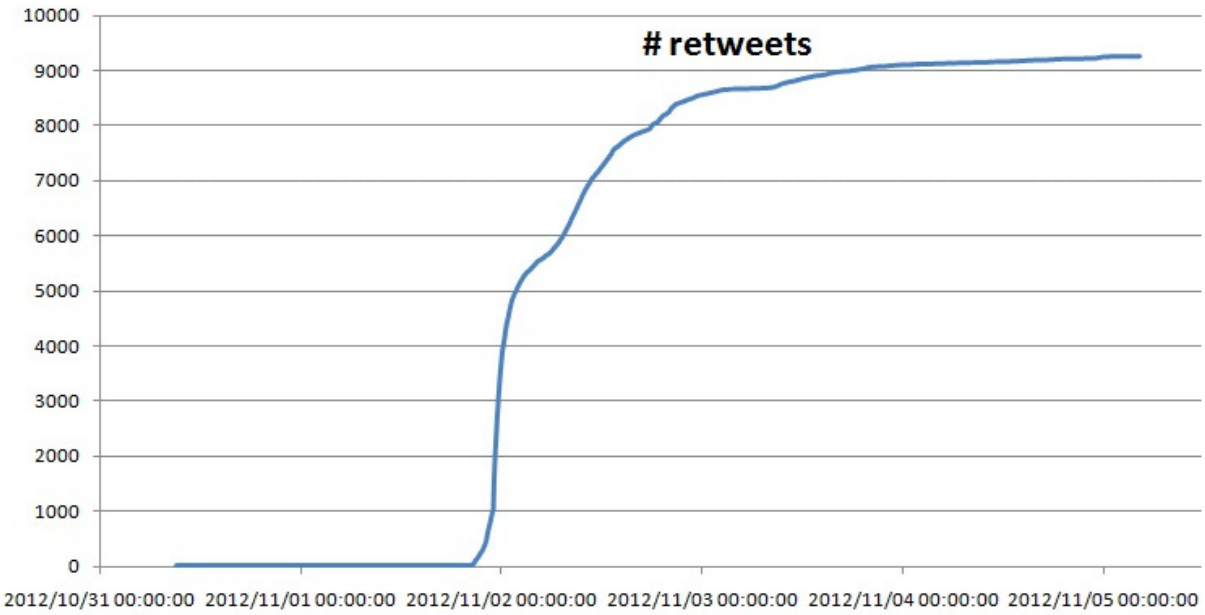
The highlights of this sheet are:

Time	# retweets	# readers
01/11/2012 20:34	3	8903
01/11/2012 21:00	103	39966
01/11/2012 22:00	354	166970
01/11/2012 23:00	989	762108
02/11/2012 0:00	3664	3103556
02/11/2012 1:00	4642	3673113
02/11/2012 2:00	5075	3765126

It's obvious that the quantum leap in retweeting was done in the hour between Nov. 1, 11:00PM and Nov. 2, 00:00 Eastern time.

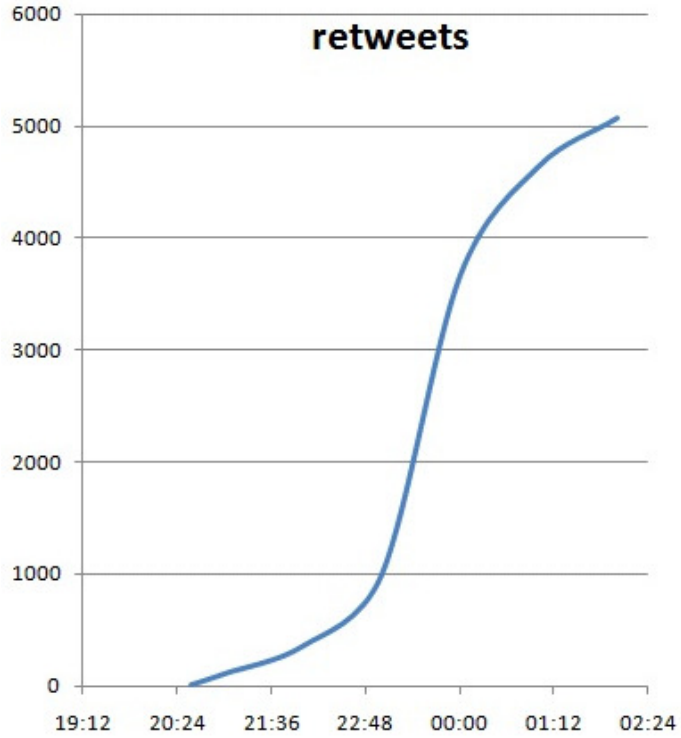
The graphs from this sheet are:

Accumulated retweets over time



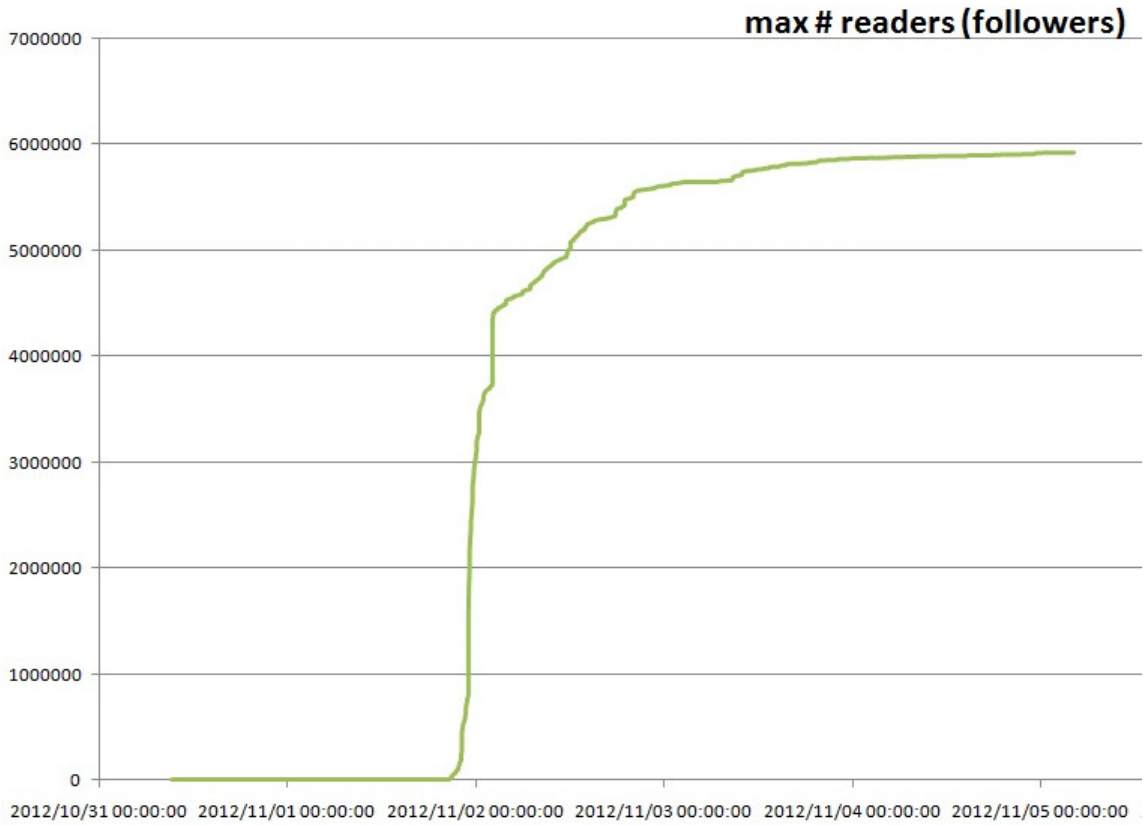
https://github.com/tsipo/twitternews/blob/master/analysis/retweets.jpg

Zoom on the interesting hours Nov. 1 – Nov. 2



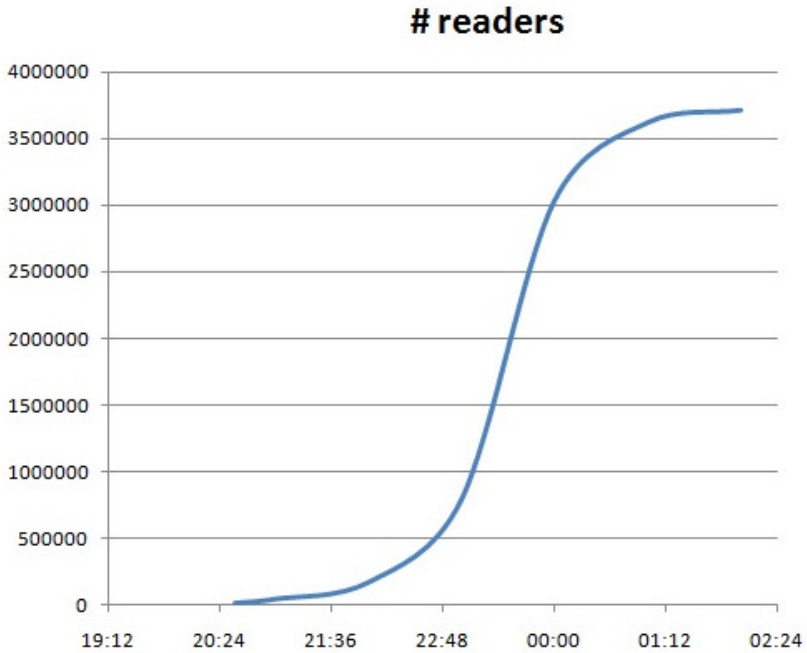
https://github.com/tsipo/twitternews/blob/master/analysis/retweets-zoom.jpg

Accumulated supremum number of readers over time



https://github.com/tsipo/twitternews/blob/master/analysis/followers.jpg

Zoom on the interesting hours Nov. 1 – Nov. 2



https://github.com/tsipo/twitternews/blob/master/analysis/followers-zoom.jpg

Analysis – influential retweeters

In its page for the tweet (http://topsy.com/twitter.com/GovChristieNJ/status/263629680731635712), Topsy indicated on Nov. 5, 4:35 AM, that it was retweeted by 346 influential people (this number has grown since to 354). But their "influential" is derived from the general Twitter status (number of followers, tweets, and retweets) rather than by their impact on the tweet in question.
Creating the graph with the assumption that the retweet was triggered by the retweeted closest to it in time, and measuring the outdegree of all retweeters, yielded the following result – which is the highlight of this entire research:
https://github.com/tsipo/twitternews/blob/master/analysis/influential-users.csv

The top of this list are the following:

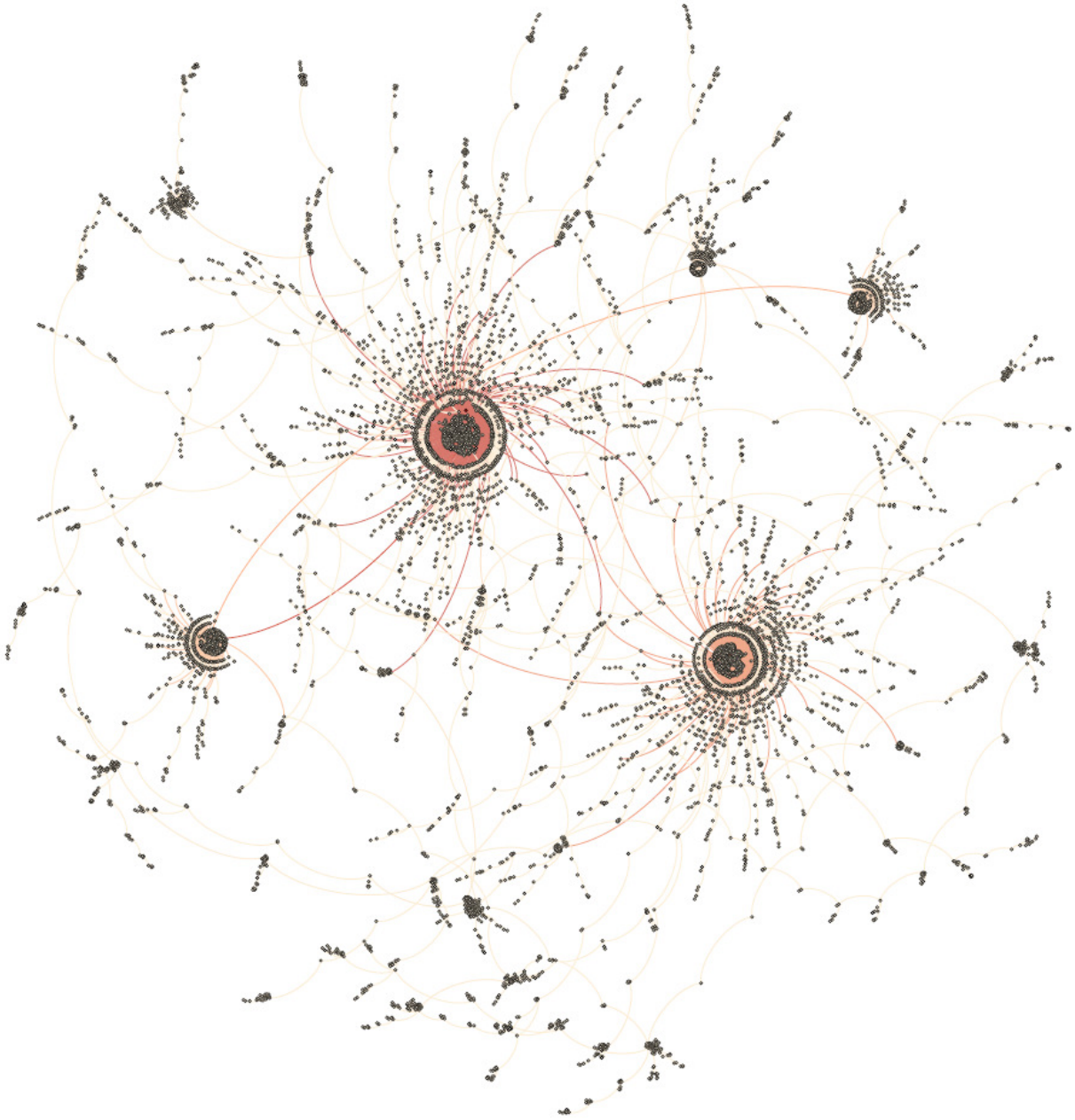
name	# of triggered retweets	# followers	retweet date	who?
ebertchicago	1884	748113	01/11/2012 23:04	Roger Ebert, film critic, Chicago, IL
unknown	1213	0	31/10/2012 9:13	-
simonhelberg	625	397700	01/11/2012 23:25	Simon Helberg, actor
markjakejohnson	426	142683	01/11/2012 23:36	Jake Johnson, actor, Los Angeles, CA
BryanVoltaggio	192	172762	01/11/2012 22:14	Bryan Voltaggio, chef, Frederik, MD
gailsimmons	125	180780	02/11/2012 0:26	Gail Simmons, culinary expert, TV celebrity
Mike_FTW	72	33793	01/11/2012 23:44	Mike Monteiro, San Francisco, CA
Sarcasticluther	49	7849	02/11/2012 2:47	Nadia Bolz-Weber, christian writer, Denver, CO
ItsAmeriie	47	618806	02/11/2012 2:08	Ameriie, singer / actress

Surprisingly, our UNKNOWN node has not won the race! :) We can see that it is responsible for 13.08% of the retweets (1231/9270). But the well-known film critic Roger Ebert has beaten it with 20.32% (1884/9270).
We can also see that most of the top influential retweeters have tweeted in the quantum leap hour (Ebert, Helberg, Johnson, Monteiro). And Ebert, who retweeted at the beginning of that hour, is probably single-handedly responsible for most of the retweets in that hour. Voltaggio in the hour before, Simmons in the hour after and Bolz-Weber and Ameriie between 2:00-3:00 AM. All of them in the night of Nov. 1 (spread over the continent's timezones).
(Almost) all of the top influential retweeters have verified accounts, which is not surprising for people with tens or hundreds of thousands of followers. We can see that this list of top influencers is composed mainly of people in the film and TV industry (or critics thereof).

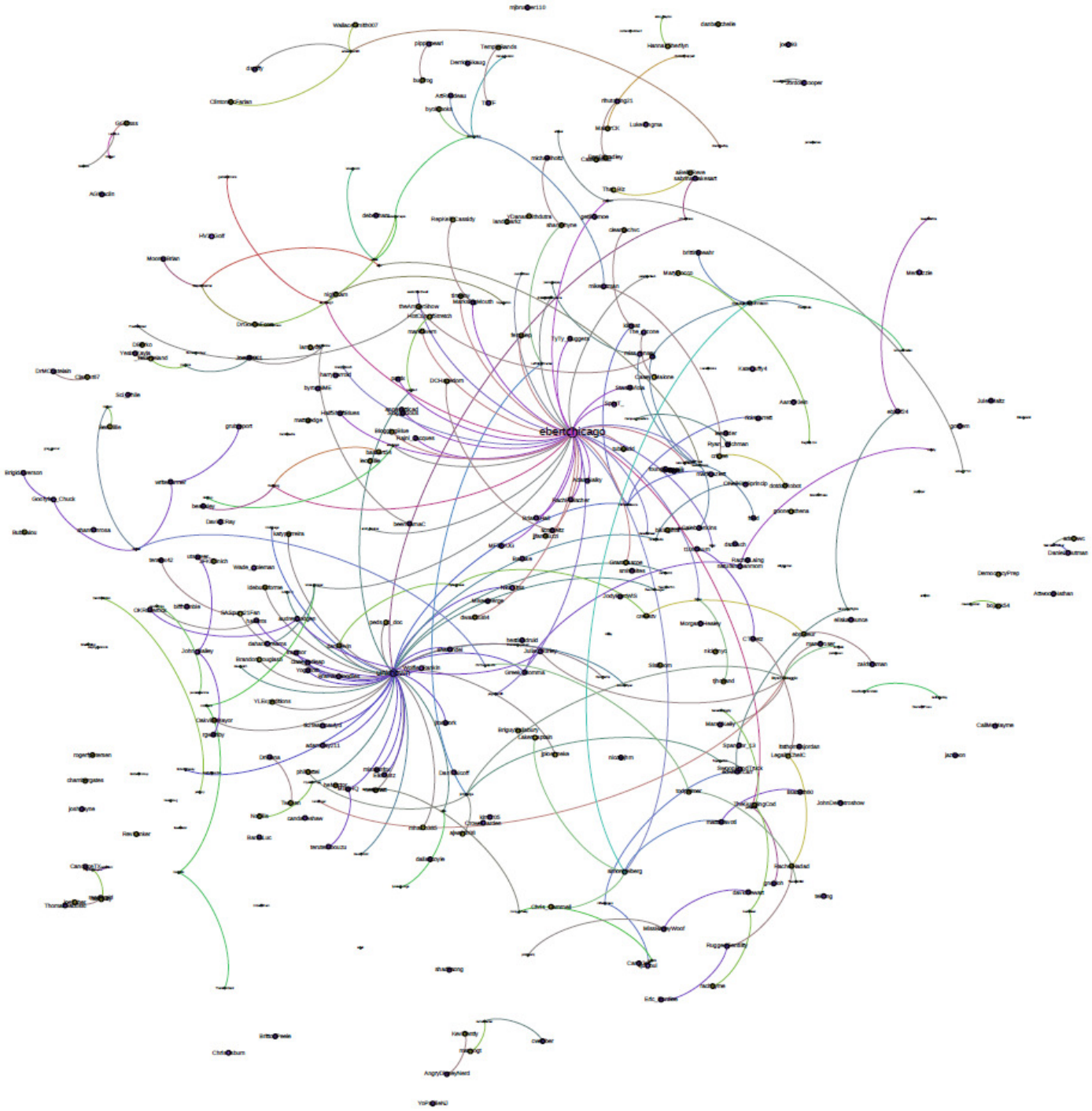
Conclusion: celebrities not only generate news, in the social media era they also spread news. Including news which they are not involved in, as in this case. Including news which are not true (remember, this is an impostor's tweet!).

The graph was then exported to GML and analyzed further in Gephi (GML file is available at https://github.com/tsipo/twitternews/blob/master/analysis/retweet.gml). The Gephi outdegree calculations match those done by my code.

The graph visualization is available at https://github.com/tsipo/twitternews/blob/master/analysis/retweets.pdf. We can clearly see the larger hubs of Ebert, UNKNOWN, Helberg and Johnson.



As this graph is not easy to work with, I have filtered with Gephi all of the nodes with degree < 4. This leaves a more viewable graph (361 nodes and 221 edges), though it is skewed – when filtering out nodes, you filter their edges too. So the total degree of the main influencers changes as well. Ebert, UNKNOWN and Helberg still kept their relative strength, but Mark Jake Johnson degree has shrunk below that of retweeters who were behind him in the original graph.



This graph is better viewed on https://github.com/tsipo/twitternews/blob/master/analysis/retweets-filtered.pdf.

EigenVector centrality

Run in Gephi for an undirected graph, assesses the outdegree results.

Name	EigenVector Centrailty
Ebertchicago	1
Unknown	0.494
Simonhelberg	0.22
Markjakejohnson	0.125
BryanVoltaggio	0.048
Gailsimmons	0.03

Community finding

Run in Gephi:

With resolution 0.1

Modularity: 0.845
Modularity with resolution: 0.036
Number of Communities: 934

With resolution 1.0:

Modularity: 0.881
Modularity with resolution: 0.881
Number of Communities: 299

Page Rank

This algorithm is absolutely not relevant to this graph and therefore was skipped.

Analysis – shortest path

Calculated by my code, due to the special structure of the graph. First, we only need paths from ORIGIN, which is the starting node of every path. Second, we need to exclude the artificial edge from ORIGIN to UNKNOWN which affects the path calculation – not the weighted one (as this edge has weight of 0.0), but of the unweighted one.
The path analysis is available at https://github.com/tsipo/twitternews/blob/master/analysis/paths.txt.

The more interesting is the average chain length (unweighted shortest path). This is the average of how many retweets were triggered until it was retweeted by each and every retwitter.
Average chain length: 6.460949298813376
Maximum chain length (graph diameter): 25

The weighted shortest path is simply the time elapsed from the original tweet until every tweet. It is affected by the cutoff time of Nov. 5, 4:15 AM. However, the bulk of retweets were done by that time

(since then until Friday, Nov. 9, 11PM only 287 retweets were added, though all of these have a very long weighted path).
Average duration: 164976.58 seconds = 1 day, 21 hours, 49 minutes and 36.58 seconds
Maximum duration is determined totally by the cutoff time and the last tweet collected:
417697.0 seconds = 4 days, 20 hours, 1 minute and 37 seconds.

References

[1] The Pulse of News in Social Media: Forecasting Popularity
http://arxiv.org/abs/1202.0332
http://www.technologyreview.com/view/426818/how-to-predict-the-spread-of-news-on-twitter/

[2] Predicting the Future with Social Media
http://arxiv.org/abs/1003.5699
https://www.technologyreview.com/view/418279/twitter-used-to-predict-box-office-revenues/