Tracking Tweets and their Retweets through Twitter

Howon Lee, Dilli Paudel, Joseph Victor

May 22, 2014

Contents

1 Introduction

When does a tweet get popular? When do its retweets echo far throughout the twittersphere, and when does it fall completely flat? Can you predict a winner seconds after its creation? How about minutes? How about before it was created at all? In this project, we investigate this problem. In particular: given a minute of information about a tweet and information about the poster, can you predict the number of retweets it gets after an hour?

To this end, we intend to build interesting and useful features for these predictions, and use standard learning algorithms.

1.1 Our Features

We propose two novel measures of Twitter retweetability, and use these to build features to learn on. The first is a `GraphRank` based algorithm called `RetweetRank`, based on work by [WLJH10]. For each user, we compute a retweetability rank, with the standard assumption that a user has a high propensity for being retweeted if they are retweeted by users with a high propensity to be retweeted.

The second measure comes from assuming a generative process through which tweets propagate through the network. Each tweet has a quality and each user has some probability of retweeting a perfect tweet of a given topic. We learn these on a body of work, and when a new tweet comes along we can learn its quality after observing it for a short period of time. The idea: if you've been retweeted by someone who seldom retweets, this is a better indicator of your eventual success than if you're retweeted by someone who retweets very often.

1.1.1 Use of Topics

In each of these measures, we use the content to separate users and tweets into topics. People don't retweet everything uniformly: a great tweet about sports is unlikely to get much attention from someone who isn't a sports fan. By allowing tweets to get popular for different reasons, we greatly increase the richness of the space of models. Tweets can get recognized for propagating through subgraphs where the strength of a specific topic is strong, rather than punishing tweets who do not appeal to everybody or users that do not enjoy a wide variety of tweets.

1.2 Our Data

For this task, we have been given a 10% sample of the Twitter Firehose for the month of March, 2013. This data, from the Twitter API, is one JSON object per tweet or retweet. It includes the tweet itself, the number of retweets and favorites at the time of the tweet (tweets have zero retweets upon first posting, of course) or retweet, information about the user (number of followers, number of friends, number of posts). For retweets, we have information about the original poster as well as about the retweeter. It is important to note that we have not been given the list of followers for a user, merely the number of followers.

2 TwitterRank and RetweetRank

2.1 Overview The TwitterRank Algorithm

Our first high powered feature is a modified TwitterRank, as in [WLJH10]. The original TwitterRank works as follows. First use LDA[1] to compute topic scores between each pair of users u and u' and each topic t, denoted $\text{sim}_t(u, u')$. Then, for each topic t, form the weighted graph where each node is a user there is a directed edge between u and u' if u follows u'. The graph weights, w_t^{TR}, are given by

$$w_t^{TR}(u, u') = \frac{T_{u'}}{\sum_{a:u\to a} T_a} \cdot \text{sim}_t(u, u') \tag{1}$$

where T_a is the number of tweets posted by user a and $u \to a$ is the binary variable indicating u follows a. For each t, we run weighted GraphRank, where with teleportation probabilities given by normalized topic strength of topic t, $\gamma_{u,t}$. The resulting rank is the t-specific TwitterRank, denoted $TR_{u,t}$.

Algorithm 1 Topic Specific TwitterRank

Use LDA to compute $\gamma_{u,t}$ and $\text{sim}_t(u, u')$ for each topic t and user pair u, u'.
for $t \in$ Topics **do**
  Initialize $TR_{u,t}$ to 1 for each user u.
  while Not converged **do**
    for $u \in$ Users **do**
      $TR_{u,t} \leftarrow \gamma_{u,t}(1-\beta) + \beta w_t^{TR}(u', u) \cdot TR_{u',t}$
    end for
  end while
end for

Intuitively, the edge weight $w_t(u, u')$ is what fraction of u's twitter feed u' takes up times their topic similarity. That is, user u, when randomly viewing his twitter feed, will randomly pick someone, but weighted such that they will topic pick a user who's topic interests them. Considering this as a random surfer is then slightly far-fetched, since the user does not then get to see his friend's feed, but we do it anyway. The idea is that users with more influence in a specific topic will dominate the feeds of users with high influence in that same topic, and this captures that notion exactly.

2.2 The RetweetRank Algorithm

The RetweetRank gives a high powered feature of the user which measures their influence in terms of accumulating retweets. The idea, if you are retweeted by someone who's tweets have a high retweet rank, then you ought to have a powerful propensity for making your tweets visible as well. In the fable of the random surfer, the tweet itself can be though of as surfing.

We modify this algorithm for our purposes in two slightly different ways. First, we consider a slightly different graph; the retweet graph, where each user is a node and there is an edge

[1] We used the package Mr.LDA to compute LDA through Variational Inference on Map Reduce [ZBGA12].

between u and u' if u has ever retweeted u'. This is both stronger and weaker than the follower graph; since one user can follow another without ever retweeting them, and one user can retweet another without following them. The latter happens in practice quite often, as a "retweet of a retweet" is indistinguishable from a single retweet. If u retweets v and w sees u's retweet and retweets it, it is only recorded that w retweeted v, and this can and does happen even if w is not following v at all!

Second, we modify the edge weights as follows.

$$w_t^{RR}(u, u') = \frac{R_{u,u'} T_{u'}}{\sum_a R_{u,a} T_a} \cdot \mathrm{sim}_t(u, u') \tag{2}$$

where $R_{u,u'}$ is the number of times u has retweeted u' and where the sum over a is over all users.

To compute the `RetweetRank`, $RR_{u,t}$, use Algorithm 1 with equation (1) replaced with equation (2) in the update step, with the sum now over all users who have retweeted u, not just followers.

2.3 Tweet Features from Retweet Rank

Given a brand new tweet s from user u, the RR_u^t give a strong and immidiate first guess at the eventual popularity of the tweet. If, further, we wait until time p after a tweet is posted, we can compute tweet features

$$RR_{s,t}^{(p)} = \sum_{u \in \mathcal{R}_s^{(p)}} RR_{u,t}$$

where $\mathcal{R}_s^{(p)}$ is the set of users who have retweeted s within a time p after s was posted. We compute these for a few values of p. Finally, if we wish to remove features we can compute overall `RetweetRank`,

$$RR_u = \sum_t RR_{u,t}$$

$$RR_s^{(p)} \sum_t RR_{s,t}^{(p)}$$

Note that it was still useful to have topics when computing the ranks originally, as this allows for a much richer space of models, even if we only ever use the aggregate over topics.

3 `AlphaPhi`: Features from a Generative Model

3.1 The model

We propose the following generative model for retweet behavior on twitter. Assume that each tweet s has a topic τ_s, known a priori. We posit that each tweet s has an intrinsic quality $\alpha_s \in [0, 1]$, and for each user u and topic t there is a number $\phi_{u,t}$, the probability of user u retweeting a perfect tweet of topic t, given that he has seen it. We then declare

$$\mathbb{P}[u \text{ retweets } s \mid u \text{ has seen } s] = \alpha_s \phi_{u,\tau_s} \tag{3}$$

Furthermore we assume every user sees every tweet or retweet posted by one of their friends.

Let $a_{s,u}$ be the binary variable indicating that user u has seen tweet s, and $b_{s,u}$ be the binary variable indicating that u has retweeted s. Assume, for now, that a and b can be observed directly from the data. We can now write down the likelihood function.

$$l(\alpha, \phi \mid a, b, \tau) = \sum_s \sum_u a_{s,u} \left[b_{s,u} \log \alpha_s \phi_{\tau_s} + (1 - b_{s,u}) \log(1 - \alpha_s \phi_{\tau_s}) \right] \tag{4}$$

3.2 Solving the `AlphaPhi` Problem

This likelihood (equation (4)) has a few key properties. One is that if α is fixed, the model is additive in the ϕ, and if ϕ is fixed the likelihood is additive in α. Another is that both conditional subproblems are convex, though the original problem need not be. This suggests the following algorithm.

Algorithm 2 Estimate `AlphaPhi` Parameters

```
Randomly initialize α and φ.
for t ∈ Topics do
    while Not converged do
        for u ∈ Users do
            Optimize φ_{u,t}, holding the rest of the variables fixed.
        end for
        for s ∈ Tweets with τ_s = t do
            Optimize α_s, holding the rest of the variables fixed.
        end for
    end while
end for
```

To make this practical, we create two files. One file lists, for each tweet, how many people retweeted it and the id's of the users that saw but did not retweet. The other lists, for each user, how many tweets they retweeted and the id's of the tweets they saw but did not retweet. Each line is enough information to compute the partial derivative of the likelihood with respect to the parameter being optimized. We compute this derivative at a fixed number of points (say 16) without using any additional memory, and choose the value closest to zero. Since we can compute the partial derivative of the likelihood in a streaming fashion, this algorithm uses no memory other than what is needed to store current values of the parameters. The memory requirements are made even easier since we consider only one topic at a time. On a Dell laptop with 2GB of RAM, this easily scales to a million tweets and a million users.

3.3 Numerical Experiments

One worries that the likelihood function (equation (4)) is not convex! Indeed, it is easy to find examples where the solution is undefined: for instance if there is a tweet that was never retweeted and only seen by a user that never retweets, clearly the likelihood has no unique maximum. Still; we plot on, and hope for the best.

Because this is a generative model, it is easy to generate data for experimental purposes, and in fact we might as well assume there is only one topic, since we learn the parameters for each topic separately. We start by using the R library `igraph` to generate a small-world network

(Twitter is assumed to be one such). Then, for each user u, we draw ϕ_u from a beta distribution. We also give each user a randomly selected beta-distribution of their own. For each tweet s, we randomly pick who the tweeter will be and draw α_s from that user's beta distribution. Then, once the parameters are drawn, we simulate the model and output the retweet events.

We ran this simulation on a wide varieties of graph sizes, connectivity levels and parameter generating hyper-parameters, and discovered that, while the problem was technically non-convex, we always found the same optimum regardless of starting point. Better yet, the parameters learned were correct: the RMSE over the parameter space was near the theoretical minimum (due to the binning) for even modestly sized problems of several hundred tweets.

3.4 Inferring the Twitter Graph

There is one problem with this model: we need Twitter's follower graph. Without it, we do not know who has and has not seen a tweet, which is required for the defining equation (3). Using the retweet graph does not work, as this dramatically overestimates the number of edges in the follower graph. In particular, a user with relatively few followers who had one tweet go viral will see the rest of his tweets punished, as the actual visibility of the later tweet will be much less than estimated by the retweet graph.

Thus we wish to estimate the probability that, for users u and u', that u follows u'. The estimate does not have to be perfectly accurate, but cheap and relatively robust. Thus, we use a naive bayes assumption, and get

$$\mathbb{P}\left[u \to u' \mid R_{u,u'} \geq r\right] = \frac{\mathbb{P}\left[u \to u'\right]\mathbb{P}\left[R_{u,u'} \geq r \mid u \to u'\right]}{\mathbb{P}\left[R_{u,u'} \geq r\right]}$$

where $u \to u'$ is a binary random variable indicating u follows u', and $R_{u,u'}$ is the number of times u has retweeted u'. We know the prior exactly:

$$\mathbb{P}\left[u \to u'\right] = \frac{F_{u'}}{N}$$

where N is the size of the Twitterati and $F_{u'}$ is the number of followers of user u'. We can estimate $\mathbb{P}\left[R_{u,u'} \geq r\right]$ directly from the data by building histograms:

$$\mathbb{P}\left[R_{u,u'} \geq r\right] = \frac{\#\{a : R_{a,u'} > r\}}{N}$$

The hard part is estimating the likelihood. For this we make additional, very strong, known-to-be-incorrect assumptions. We assume that the retweets of a user u' are distributed uniformly among their $F_{u'}$ followers. Thus $R_{u,u'}|(u \to u')$ is distributed as binomial with probability $1/F_{u'}$ and $\sum_a R_{a,u'}$ trials. Even if this assumption were true, it will lead to low-biased estimates of the posterior probability, since some retweets are made by non-followers, and since the assumption is wrong, for the users that don't have most of the retweets, the estimates will be low-biased. This is actually a good thing: at the very least we won't dramatically overestimate the number of edges while still counting the most adamant followers.

As a sanity check, if $R_{u,u'} = 0$, this should be approximately zero. Indeed it is, as the prior is always approximately zero and normalizer will be approximately 1, thus making the posterior probability extremely small regardless of the likelihood. For this reason, if $R_{u,u'} = 0$, we set this probability to zero.

Now, for any user u and tweet s, we have

$$\begin{aligned}\mathbb{P}[u \text{ retweets } s] &= \mathbb{P}[u \text{ has seen } s]\,\alpha_s\phi_{u,\tau_s} \\ &= \left(1 - \prod_{u':b_{s,u'}=1} (1 - \mathbb{P}[u \to u' \mid R_{u,u'}])\right)\alpha_s\phi_{u,\tau_s} \end{aligned} \tag{5}$$

where $b_{s,u'}$ indicates u' retweeting s, as in equation (4). We can still optimize individual parameters in Algorithm 2 with only a slight modification; they key properties of the likelihood function still hold.

3.5 Using the Learned Parameters on New Tweets

Once the parameters α and ϕ are learned, α can be discarded. When a new tweet s is observed at time p after its creation, giving early observations of its retweet behavior, we can (easily) compute estimate $\alpha_s^{(p)}$. We use values of $\alpha_s^{(p)}$ for a few small values of p as high powered features to predict the eventual popularity of s.

4 Building a Model

5 Results

References

[CAD+14] Justin Cheng, Lada A. Adamic, P. Alex Dow, Jon M. Kleinberg, and Jure Leskovec. Can cascades be predicted? *CoRR*, abs/1403.4608, 2014.

[WLJH10] Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. Twitterrank: Finding topic-sensitive influential twitterers. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 261–270, New York, NY, USA, 2010. ACM.

[ZBGA12] Ke Zhai, Jordan L. Boyd-Graber, Nima Asadi 0001, and Mohamad L. Alkhouja. Mr. lda: a flexible large scale topic modeling package using variational inference in mapreduce. In Alain Mille, Fabien L. Gandon, Jacques Misselis, Michael Rabinovich, and Steffen Staab, editors, *WWW*, pages 879–888. ACM, 2012.