Tracking Tweets and their Retweets through Twitter

Howon Lee, Dilli Paudel, Joseph Victor

June 10, 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 few minutes 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 types of novel measures of Twitter retweetability, and use these to build features to learn on. The first are based on a `GraphRank`algorithm called `RetweetRank`, based on a similar measure called `TwitterRank`, from [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. We compute an intrinsic measure of the quality of a tweet and build features for predicting number of retweets. 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 described 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 $\mathrm{sim}_t(u, u')$. Then, for each topic t, form the weighted graph where each node is a user and there is a directed edge between u and u' if u follows u'. The graph weights, w_t^{TR}, are given by

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

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 $TR_{u,t}$.

Intuitively, the edge weight $w_t(u, u')$ is what fraction of us 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 be seen frequently by users with high influence in that same topic, and this captures that notion exactly.

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

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
```

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 ways. First, we consider a slightly different graph; the retweet graph, where each user is a node and there is an edge 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.

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

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 "partial `TwitterRank`" features as follows:

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

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`,

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

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

The $RR_s^{(p)}$ features should be thought of as a fancy version of the retweet rate, since it is just a weighted sum over the $\mathcal{R}_u^{(p)}$. We graph it against the eventual number of retweets. As you can see in Figure 1b, there is a correlation between the eventual number of retweets and the partial retweet rank. 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.

The RR_u features follow, not surprisingly, a power-law-like distribution, see Figure 1a. Most users have very small `RetweetRank`, but a handful have extremely large ranks. We take logs to make this feature manageable.

3 AlphaPhi: A Generative Model for learning a Tweet's Intrinsic Quality

For many of us, a tweet is considered a success if we get just a few retweets, since we have just a few followers. Conversely, some people can get many retweets and consider the tweet a failure, because they have very many of followers and expect more. We propose the following model that allows us to measure the intrinsic quality of a tweet independent of its visibility.

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

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

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 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 $\alpha$ and $\phi$.
for $t \in$ Topics do
    while Not converged do
        for $u \in$ Users do
            Optimize $\phi_{u,t}$, holding the rest of the variables fixed.
        end for
        for $s \in$ Tweets with $\tau_s = t$ do
            Optimize $\alpha_s$, 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 plod 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:

$$\mathbb{P}\left[R_{u,u'} \geq r\right] = \frac{\#\{a : R_{a,u'} \geq r\}}{N-1}$$

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 $R'_u := \sum_a R_{a,u'}$ 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.

We get

$$\begin{aligned}\mathbb{P}\left[u \to u' \mid R_{u,u'} \geq r\right] &= \frac{\mathbb{P}\left[u \to u'\right]\mathbb{P}\left[R_{u,u'} \geq r \mid u \to u'\right]}{\mathbb{P}\left[R_{u,u'} \geq r\right]} \\ &= \left(\frac{F_{u'}}{N}\right)\left(\frac{N-1}{\#\{a : R_{a,u'} \geq r\}}\right)\mathbb{P}\left[\text{Binomial}(1/F_{u'}, R_{u'}) \geq r\right] \qquad (5)\end{aligned}$$

As a sanity check, if $r = 0$, 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 for sparsity's sake.

Further sanity checking, if $r = 1$, for users with few retweets and a reasonable number of followers (average, non celebrity users), this will be high (possibly greater than 1, so we need to clamp!), which is good since we don't expect many retweets from non-followers, so even one retweet means you probably follow that person. If $F_{u'} = 1$ and only one person retweets, that one person is a follower with probability 1, since the variance in the binomial is zero! Similar statements remain true if the number of re-tweeters is approximately the number of followers, so for average users this approximation is probably reasonable.

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

$$\begin{aligned} \mathbb{P}[u \text{ retweets } s] &= \mathbb{P}[u \text{ has seen } s]\, \alpha_s \phi_{u,\tau_s} \\ &= \left(1 - \prod_{u' : b_{s,u'} = 1} (1 - \mathbb{P}[u \to u' \mid R_{u,u'}])\right) \alpha_s \phi_{u,\tau_s} \end{aligned} \tag{6}$$

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 Features from `AlphaPhi`: Partial Alphas

The α_s parameters are, in and of themselves, an interesting feature, since they attempt to tell us how good a tweet is independent of a user's influence. However, we still wish to predict the total number of eventual retweets, so we ask for a little bit more. Thus we propose the following predictive features based on this model.

When a new tweet s is observed at time p after its creation, giving early observations of its retweet behavior, we can (easily) estimate alpha as if its complete behavior was known, denoted $\alpha_s^{(p)}$, which we call the "partial alpha" at time p. In contrast we call α_s the "total alpha". 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.

Another useful feature, given a tweet s of topic t at time p after its creation, is the average value of the $\phi_{u,t}$ for each u that retweeted s so far. We call this feature $\phi_s^{(p)}$, and it ended up doing pretty well in linear and random forest models (Figure 2a and Figure 2c). This feature should be thought of as a fancy version of the retweet rate itself, since it is essentially a weighted sum over the $\mathcal{R}_u^{(p)}$.

3.6 Interpreting the parameters

Not surprisingly, most of the $\phi_{u,t}$'s are zero: most users don't ever retweet things of most topics (some users rarely retweet at all). In fact, 98.8% of all the $\phi_{u,t}$ values were zero, while 1.1% were exactly one. Explaining the second fact is equally easy: some people retweet almost everything they see from a topic (recall our learning algorithm optimized by binning, so to report a $\phi_{u,t} = 1.0$ merely means its closer to 1 than 0.86). In the middle, there is a nice spread, as one might expect.

Interpreting the α parameters must be done with care, especially when trying to predict the eventual number of retweets. Indeed, and α can be high both because it was often retweeted and because it was seldom seen. Users with very few followers often produced these high quality tweets, although many of the most retweeted also had very high quality. Conversely, a tweet could be low quality because it was over-exposed. We look at the plot (Figure 1) α_s and $\alpha_s \log(F_u + 1)$ versus the $\log(R_s^{1hr} + 1)$, log of the number of retweets after an hour. The relationships are complicated, but not surprising. One notices most tweets are banded along $\alpha_s = 0$ and number $R_s^{1hr} \leq 3$. Outside of these "most tweets are boring" bands, we see what looks like a positive correlation between $\alpha_s \log(F_u + 1)$ and the number of retweets, as we should expect; users with many followers and high quality tweets get many retweets. However, even very retweeted tweets can have low alpha as they are overly visible among people with positive phi's in that topic.

4 Experimental Setup and Results

4.1 Feature Computation

We extracted a topic model using 30-topic LDA for all 3,296,605 users active between March 1, 2014 and March 5, 2014, and went on to compute their `RetweetRank` using all tweets and retweets posted during this period.

From this period, a random sample of 10,000 tweets were selected, on which `AlphaPhi` was computed using the twitter graph inferred during this time and all 93,151 users inferred to have seen these 10,000 tweets. We also, for a period of 1 hour after the creation of each tweet, record $R_s^{(p)}$ every twenty seconds. Finally, for the first ten minutes after each tweet is posted, we compute $\alpha_s^{(p)}$, $\phi_s^{(p)}$ and $RR_s^{(p)}$ in twenty second intervals, using the ϕ's computed before.

4.2 The harder problem: Predicting the number of retweets after an hour from data at time p.

It is a very difficult problem to predict very heavy-tailed distributions where most tweets have very few retweets, but some small number have very many. We attempt it anyways. For each minute p in the first ten minutes, we build a machine learning model to predict $\log(R_s^{1hr} + 1)$ from the predictors available at time p. We report the 5 fold cross validation with two repeats for each model and report the root mean squared error in the predicted logarithm. The machine learning models we use are linear regression, support vector regressions and random forests (all untuned because of the long time it takes to fit the number of models we want to fit).

In each run we use a different feature set described in this paper, plus ambient features logs of the friend count and followers count.

4.3 The Doable Problem: Predicting Cascades

As described in [CAD+14], there is a brilliant way of overcoming the difficulty of heavy tails. Balanced classification problems are inherently easier than highly unbalanced regressions. Luckily, because the number of retweets a tweet eventually gets ought to follow a power law, the probability of a tweet doubling its retweet count is about 50-50. Thus, for each $i \in \{1, ..., 8\}$ we predict, using a support vector machine, if a tweet with at least 2^i retweets will reach 2^{i+1} retweets.

The features we use for this prediction are the ambient features from before, plus

$$p_{s,i} = min p \ : \ R_s^{(p)} \geq 2^i$$

and features $RT_s^{(p_{s,i})}$, $\alpha_s^{(p_{s,i})}$ and $\phi_s^{(p_{s,i})}$. We also experiment with using α_s itself for prediction, although this is impossible in real applications due to it not yet being known.

5 Results

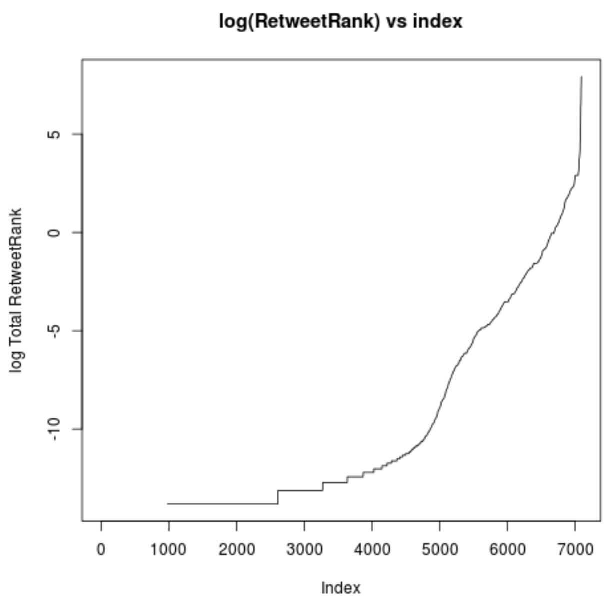
5.1 Results for the Regression Problem

The results for regression are summarized in Figure 2a, Figure 2b, Figure 2c. In each graph we include the constant predictor (predict using only things known at tweet creation: followers and friends count), and in almost ever case, we beat it.

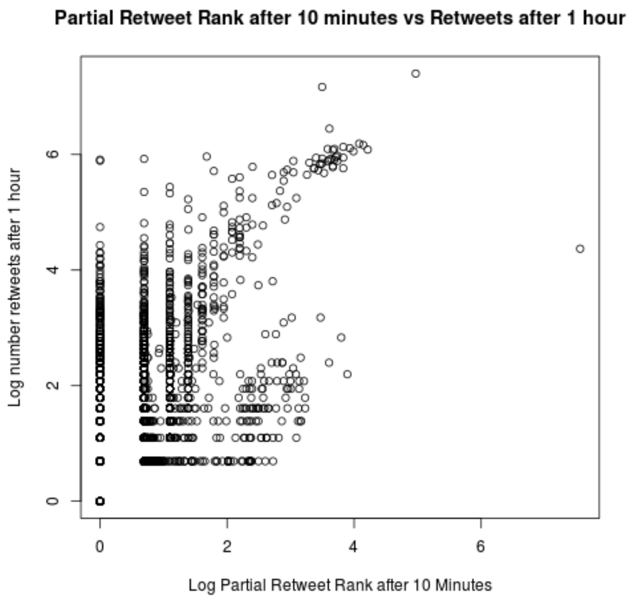
Without fancy features, the naive first try would be to just use logs of $R_s^{(p)}$: this is the thing to beat. We look first at the linear model. Notice first of all that most features sets which do not use $R_s^{(p)}$ do much worse; the exception being linear models on the partial `RetweetRank` (Figure 2a orange). This is sad, but not surprising, as these things try to encode fancy information not directly related to what we are computing, while the retwet rates are literally early predictors of this response, and at least they beat the constant estimator. The remaining linear models on features sets including the retweet rates do almost exactly as well as just the retweet rates, as the three nearly identical curves in the middle of Figure 2a shows.

On the SVM, the best feature set is $\alpha_s^{(p)} \log(F_u + 1)$ with the retweet rates (Figure 2b black). It is not surprising a nonlinear fit is required to make this feature set really shine; the partial alphas can go up and down for all sorts of complicated reasons, but apparently the SVM could sort it out. The partial alphas by themselves also did well (purple). The other feature sets did worse than the most naive model (red), although still better than they did with the linear model.

On Random Forests, we beat the naive when we either add the partial alphas or the partial retweet ranks, and both do about as well as the other. The phi averages do no worse than the naive, while the features which do not use the retweet rates do worse than the naive approach, but still better than they did in the linear models.



(a) The log of the RR_u plotting against its sorted index.



(b) The relation between $RR_u^{(10min)}$ and R_u^{1hr}

5.2 Results for the Cascade Prediction Problem

The results for this approach, summarized in 2d are far more positive. The baseline approach, the approach-to-beat, is only using the rewtweet rates. Excitingly, adding partial alphas (red and blue) beat the baseline! Knowing the total α_s (red) helps a bit early on, although it seems to have diminishing returns for larger tweets. Sadly, the other two features both performed no better than baseline.

6 Difficulties and Future Work

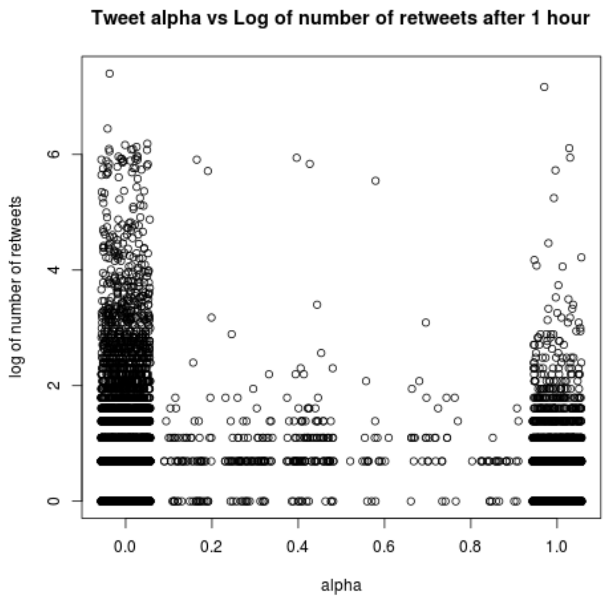
One major difficulty we had was the massive pipeline required to compute the features we cared about. In particular, computing the LDA model and tweet topic took days of compute time and required careful use of AWS resources. Many things about AWS were found to be non-intuitive, like `HIVE` require explicit copying of the data from `S3` to run in any reasonable amount of time and seeming to be very sensitive to the way queries were written. The long running jobs all dependent on each other were insanity inducing, and we barely finished on time.

Another difficulty is the incomplete data. While the retweet rates were known almost exactly (since we could read the number of retweets from the JSON file), the other features were build on looking at the individual rewteeters, of whom we knew only 10%. This made those features far less robust than the retweet rate itself. It would be interesting to see if using the entire firehose would improve performance. Estimates of the graph were likewise annoying, and we wonder if we could build better `AlphaPhi` estimate if we knew the exact graph.

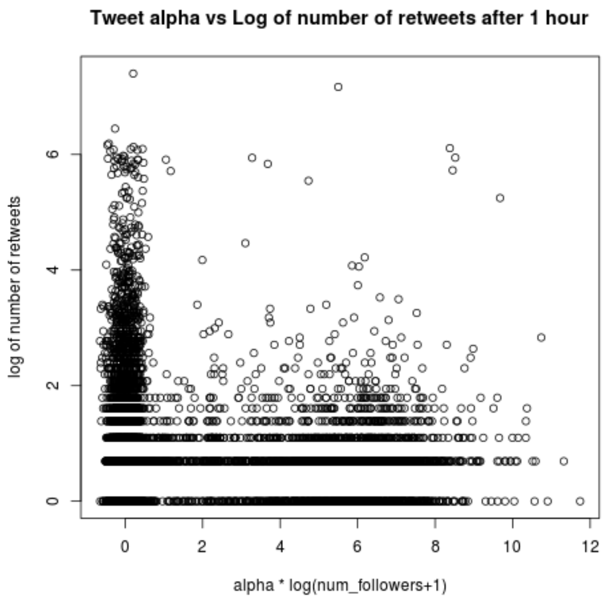
It would also be interesting to try playing with the number of topics, combining classes of features and tuning machine learning algorithms for better performance.

7 Conclusions

We created two new sets of measures, one of user retweetability and one of tweet quality, and used them to engineer features to predict the number of retweets. Our initial results with `AlphaPhi` were promising, we were able to improve both regressions and cascade predictions, although they require a nonlinear algorithm to unlock their full potential. The `TwitterRanke` was very predictive with linear predictors, not requiring non linearity at all. We were able to do even better when trying to predict cascades. Predicting tweet popularity is a difficult problem, and I think we gave it a good fight.



(c) Total Alpha vs log Retweets



(d) Total Alpha * log of followers vs log Retweets

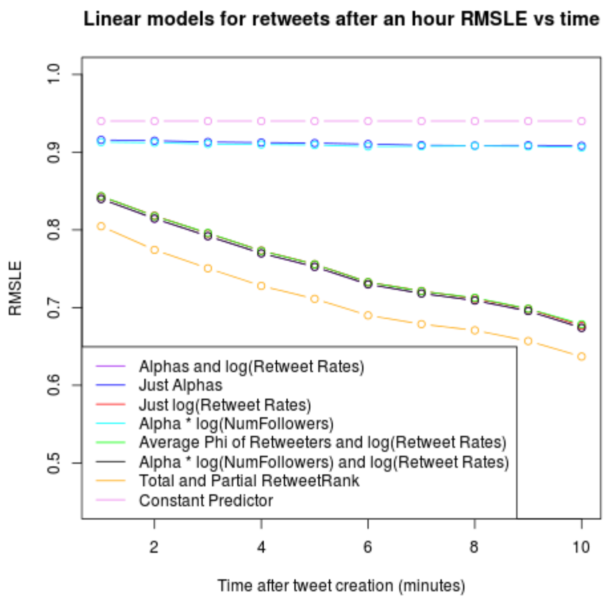
Figure 1: Relation between Alpha and log of Retweets after an hour)

References

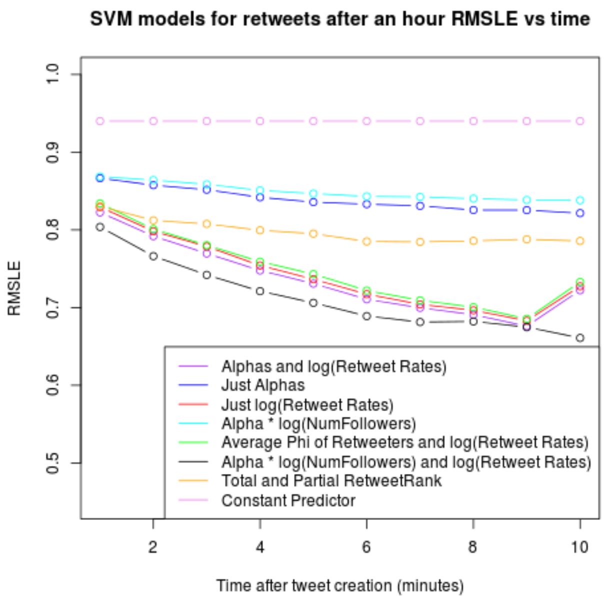
[CAD+14] Justin Cheng, Lada A. Adamic, P. Alex Dow, Jon M. Kleinberg, and Jure Leskovec. Can cascades be predicted? *CoRR*, abs/1403.4608, 2014.

[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.

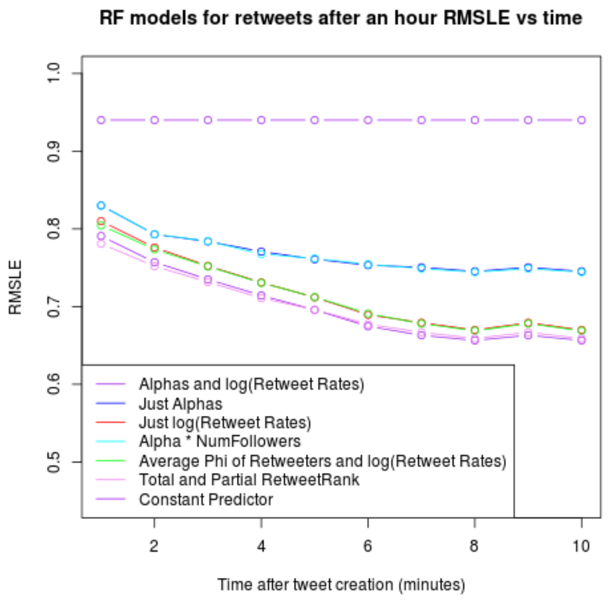
[ZBGA12] Ke Zhai, Jordan L. Boyd-Graber, Nima Asadi 0001, and Mohamad L. Alkhouja. Mr. lda: a flexible large scale topic modeling package using variational inference in mapreduce. In Alain Mille, Fabien L. Gandon, Jacques Misselis, Michael Rabinovich, and Steffen Staab, editors, *WWW*, pages 879–888.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



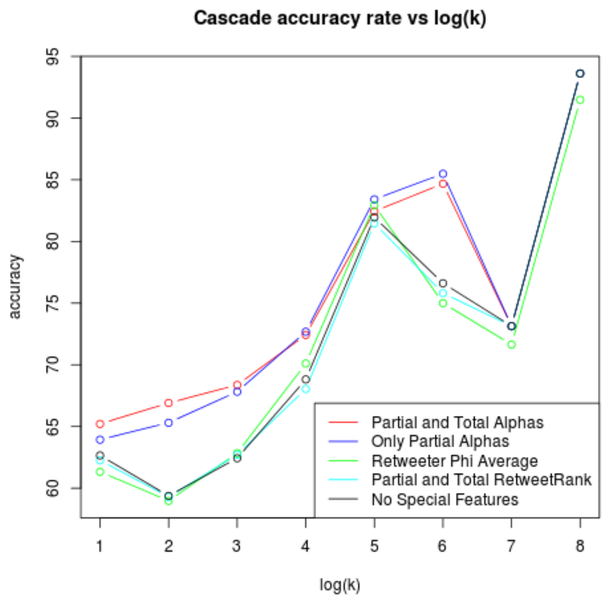
(a) Early prediction curves for various feature sets using Linear Models



(b) Early prediction curves for various feature sets using Support Vector Machines



(c) Early prediction curves for various feature sets using Random Forests



(d) Accuracy rates using “cascading” experimental setup from [CAD+14]

Figure 2: Experimental Prediction Results