Outtweeting the Twitterers - Predicting Information Cascades in Microblogs

Wojciech Galuba, Karl Aberer
EPFL, Switzerland

Dipanjan Chakraborty
IBM Research, India

Zoran Despotovic, Wolfgang Kellerer
DOCOMO Euro-Labs, Munich, Germany

Abstract

Microblogging sites are a unique and dynamic Web 2.0 communication medium. Understanding the information flow in these systems can not only provide better insights into the underlying sociology, but is also crucial for applications such as content ranking, recommendation and filtering, spam detection and viral marketing. In this paper, we characterize the propagation of URLs in the social network of Twitter, a popular microblogging site. We track 15 million URLs exchanged among 2.7 million users over a 300 hour period. Data analysis uncovers several statistical regularities in the user activity, the social graph, the structure of the URL cascades and the communication dynamics. Based on these results we propose a propagation model that predicts which users are likely to mention which URLs. The model correctly accounts for more than half of the URL mentions in our data set, while maintaining a false positive rate lower than 15%.

1 Introduction

Microblogging is a relatively new phenomenon in the Web 2.0 world of user generated content. Twitter[1] is one of the most popular microblogging sites today. Twitter users post *tweets*, short messages of up to 140 characters containing a variety of content [9, 2], ranging from daily activity updates, discussions, photos, interesting URLs and random thoughts. Each Twitter user chooses which other users to follow and the tweets from the followed users are aggregated in a single reverse-chronologically ordered stream.

The social graph of followers is a unique and very dynamic communication medium and is host to an increasing number of viral phenomena: breaking news propagation, emergency broadcasts, marketing, public relations, campaigning, activism and many more. Modelling and understanding the flow of information in microblogging systems can potentially lead to more effective use of this new communication medium and provide insights into the underlying sociology.

[1] http://twitter.com

One of the commonly shared pieces of information on Twitter are the URLs. When a user tweets an interesting URL, her followers, in turn, might re-tweet it to let their followers know about it [4]. This is the basic mechanism through which the URLs spread in the follower graph.

Goal. In this paper, we focus on characterizing and modelling the information cascades formed by the individual URL mentions in the Twitter follower graph. The information cascades have been studied before in other Web 2.0 systems, such as Flickr [3], blogs [14, 17], Digg [16, 13] and YouTube [16]. Most of the prior work builds information propagation models to either reproduce cascades with statistical properties matching the empirical observations or to predict how far the information will diffuse in the network given its initial spread. We address a different problem: predicting *which users* will tweet *which URLs* given a training set of existing URL mentions.

Motivation. Accurate prediction of URL mentions is an important enabler of a number of possible applications. First, knowing the tweeting probabilities for each user and URL can be used to generate a ranked list of URLs for each user providing a *personalized recommendation* of URLs that the user is likely to find interesting. For users who follow many other users, this method can be used to prevent information overload by *ranking and filtering* the incoming tweets. Second, aggregating the probabilities per URL quantifies the URL's future potential to diffuse in the social network. This could serve as method for early identification of viral URLs. Third, having an accurate propagation model trained for a specific social network can also help *viral marketing campaigns* to select URL injection points that would maximize the spread of the campaign URL in the network. Finally, a model predicting URL diffusion is not only useful when its predictions are correct, but also when the new data does not match them. A sudden outbreak of anomalous activity that is not correctly predicted by the model most definitely signifies an event worthy of atten-

tion. This approach could potentially be used for *spam detection*.

Outline & results. In this paper, we analyze the spread of 15M (million) unique URLs among 2.7M users based on a 300 hour data set aggregated from Twitter. We measure several statistical properties of the data. First, the Twitter follower graph is a small world with a giant connected component and mean shortest path of 3.61. Second, the tweeting frequencies across the different users and across the different URLs are power-law distributed. Third, the information cascades on the social graph tend to be shallow and wide, having an exponentially distributed depth. Fourth, the cascades for each URL are composed of smaller connected components, whose both number per-cascade and size follow power-law distributions. Finally, the diffusion delay between URL tweets in a cascade is log-normally distributed with a median of 50 minutes.

Based on the above empirical observations, we propose a propagation model that simultaneously takes into account several key factors: content popularity, user influence and the rate of propagation. These factors become the unknown parameters of the model. We use the gradient ascent method to find the parameter values that maximize the number of correctly predicted URL mentions in the Twitter test data. The evaluation shows that the model can predict more than half of the individual URL mentions in the test data set while having a less than 15% false positive rate.

2 Related work

Diffusion processes in networks have been studied in a variety of different areas. In epidemiology, extensive research has been done in predicting the spread of disease in populations. A number of models draw on that research to model the spread of information in the social graph as viral processes [8, 1, 15]. However, epidemiological models generate bimodal distributions of infection sizes, i.e., either most or few nodes are infected with information. This does not explain the observations from Web 2.0 systems well, which has led to development of alternative models, e.g. in which the transmissibility falls off with distance from the source [17].

Information diffusion has been studied in many Web 2.0 systems, including Flickr [3], blogs [14, 17], Digg [16, 13] and YouTube [16]. This work, in general, focuses on building models that generate information cascades whose statistical properties match the empirical observations; or models that predict how far the information will spread in the network. Unlike the existing work, our aim is to make predictions at a much finer level of granularity per user-URL pair, rather than modelling larger-scale aggregates of user activity.

Several models for diffusion on graphs have been proposed. In the *linear threshold model* [6] each node has an associated threshold value. If the number of infected neighbors of a node exceeds its threshold then the node itself becomes infected. The *independent cascade model* [5] associates a fixed spreading probability per graph edge and allows each node to attempt infecting another node only once. Further studies have generalized these models [10]. In all of the work, the propagation model's parameters are either constant or drawn from distributions. In our approach, we find the optimal parameter values by training the model on actual information diffusion data.

Earlier measurement studies of Twitter measured the properties of the follower graph and looked at its changes over time[11, 7], others studied the geographic distribution of users [9] and examined the phenomenon of retweeting [4]. Most recently, the results of Kwak et al. [12] look a number of properties of the Twitterspehere. Many of the results align with ours, in particular, retweet trees are measured to be shallow and their with power-law distributed size.

3 Data set

3.1 Data acquisition

Tweets. For 300 hours, starting on Thu, 10 Sep 2009 19:56:47 GMT the Twitter Search API was continuously queried for the search string `http`. The text of each tweet returned by the query was parsed for any URLs and user names it contained.

URLs. Each URL mentioned in the tweets was stored. If the URL was created by one of the popular URL shortening services, HTTP redirects were recursively followed to expand the URL to its original form. All the URLs were also URL-decoded to ensure uniform representation under the percent-encoding (`%xx`) notation.

User graph. For each tweet, we queried the Twitter API for the metadata about the tweet's author as well as all the users that the author follows.

The final outcome is a dataset of timestamped URL mentions together with where they happened in the social graph.

3.2 Limitations

Stream continuity. The Twitter search API allows for specifying the minimum timestamp for the tweets returned in the query results. In this way, the complete stream of tweets can be systematically downloaded. The combined availability of Twitter and our crawling infrastructure during the data acquisition period was 99.6%,

component size	2	3	4	5	6	2483290
#components	513	31	6	2	1	1

Table 1: The size distribution of the strongly connected components of the follower graph.

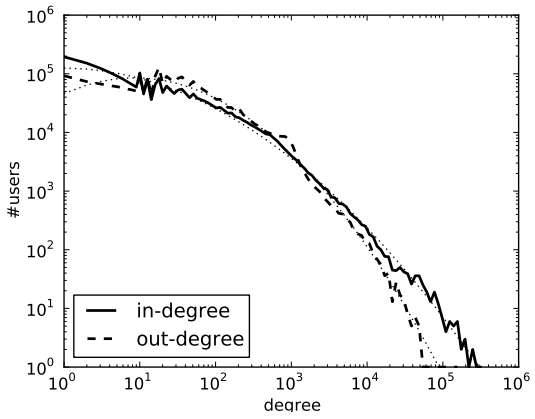

Figure 1: **User graph degree distribution.** In the follower graph, each arc points from the follower to the followee. Both the node in-degree and the out-degree distributions are log-normal (dotted lines, in-degree: $\propto \ln\mathcal{N}(\mu = 6.85, \sigma^2 = 6.55)$, out-degree: $\propto \ln\mathcal{N}(\mu = 5.99, \sigma^2 = 3.84)$))

which means that at most 0.5% of the tweets can possibly be missing from the stream.

User graph. The user graph contains only those users whose tweets appeared in the stream, i.e., only users that during the 300 hour observation period posted at least one public tweet containing a URL. The graph does not contain any users who do not mention any URLs in their tweets or users that have chosen to make their Twitter stream private.

For each newly encountered user ID, the list of followed users was only fetched once. Our data set does not capture the changes occurring in the user graph over the observation period.

4 Follower graph properties

The user graph consists of 2.7M nodes connected with 218M arcs. Each arc points from the *follower* to the *followee*.

Component sizes. The user graph is directed, the distribution of the sizes of the strongly connected component sizes is in Table 1. The Twitter user graph has a giant connected component encompassing the majority of the users. This is characteristic of many other social networks, both on-line and real-world (§2).

Degree distribution. We have measured the degree distribution in the user graph (Fig. 1). Both the in-degree and the out-degree distributions have tight log-normal fits.

Path lengths. An important structural metric in graphs is their shortest path distribution. The Twitter graph is a small world (Fig. 2) with a mean directed path length of 3.61.

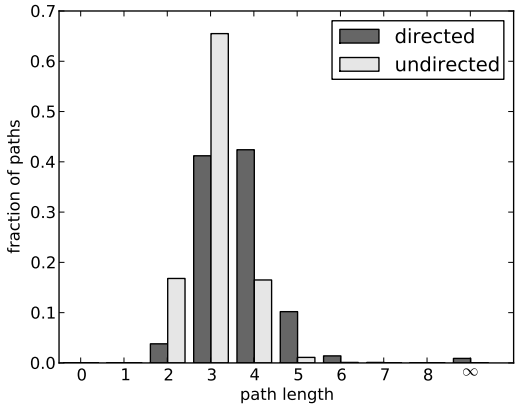

Figure 2: **Shortest path length distribution.** We compute the shortest path for each of the 1000 randomly chosen source-destination pairs. There are two cases: the paths are allowed to traverse along the arcs only (*directed*) or in both ways ignoring the direction of the arc (*undirected*). The ∞ symbol indicates that no path exists for a source-destination pair.

5 URL sharing activity

URL mentions. Users can include more than one URL as part of the tweets. In our set of 27M tweets there are 15M unique URLs mentioned. Each tweet has a maximum length of 140 characters, which puts a limit on the maximum number of URLs that can be included in a single tweet. The vast majority of the tweets contain no more than two URLs.

User activity. Users vary wildly in how frequently they tweet URLs. For the two different metrics we have used, the user activity is power-law distributed (Fig. 3).

URL popularity. The different URLs are mentioned with different frequencies in the Twitter social network, again fitting the power-law distribution. (Fig. 4).

6 Information cascades

When `@alice`, a Twitter user, mentions a URL, all of her followers can immediately see her tweet in their feeds. When `@bob`, a follower of `@alice`, sees a URL coming from her, he can also tweet that URL to let his followers know about it. This is one of the common mechanisms through which URLs propagate in the Twitter social network.

We have chosen the URLs as the subject of the study since they are a low-noise language-independent signal whose propagation can be easily tracked using the Twitter API.

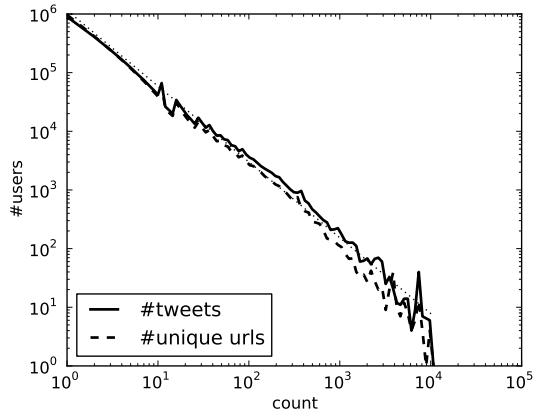

Figure 3: **User activity.** The user activity measured by how many tweets or unique URLs the user posted. For both metrics, the user activity is power-law distributed (dotted line: $\propto x^{-1.25}$).

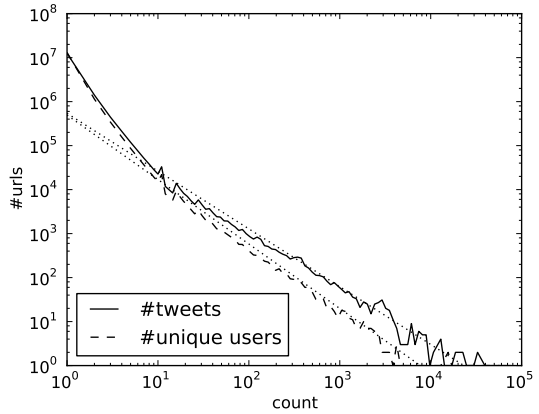

Figure 4: **URL popularity.** URL popularity measured by the number of tweets and number of unique users that mentioned the URL. In both cases we obtain the power-law distributions, $\propto x^{-1.3}$ for tweets and $\propto x^{-1.45}$ for unique users.

Retweets. Often when users tweet a URL that was found in another user's feed, they give credit to the original URL poster. This phenomenon became known as retweeting [4]. If `@bob` wants to give credit to `@alice`, he prepends her message with `RT @alice:` followed by the text of the original tweet of `@alice`. `RT` stands for re-tweet. We only focus on the tweets containing URLs, even though any tweet can be retweeted in this way.

Retweets are a strong indication of the direction of information flow in the Twitter social graph in that they explicitly identify the source.

6.1 Definitions

We next define two types of information cascades: *F-cascades*, for which the flow of URLs is constrained to the follower graph and *RT-cascades*, for which we disregard the follower graph and use the who-credits-whom data from the retweets.

Let $G(V, E)$ be the *Twitter follower graph* consisting of users from the set V. Graph G is directed, there is an arc $(v_1, v_2) \in E$ in the graph iff the user v_1 follows user v_2. Let U be the set of all URLs.

Let a ***F-cascade*** $F(u)$ be a graph of all the users that have tweeted the URL $u \in U$. An arc (v_1, v_2) exists in $F(u)$ iff: 1) v_1 and v_2 tweeted about u, 2) v_1 mentioned u before v_2 and 3) v_2 is the follower of v_1.

Similarly, the ***RT-cascade*** $R(u)$ is a graph of all the users that have either retweeted the URL $u \in U$ or have been credited as the source of the URL in a retweet. An arc (v_1, v_2) exists in $R(u)$ iff: 1) v_1 tweeted about u, 2) v_1 mentioned u before v_2 and 3) v_2 credited v_1 as the source of the URL u.

Each retweet credits only a single user[2], which makes each RT-cascade a forest consisting of trees. On the other hand, in the F-cascades, each node can have several followees that mentioned a URL before it and in general F-cascades are directed acyclic graphs. Both F- and RT-cascades are not necessarily connected graphs. Each weakly connected component of an F- or RT-cascade is a *subcascade*. The *root* of a subcascade is the subcascade node that mentioned the URL first.

6.2 Cascade properties

RT-cascades vs. F-cascades. Even though there is a large overlap between the two types of cascades, 33% of the retweets that we have observed credit users that the retweeters do not follow.

Number of subcascades. Each cascade consists of one or more subcascades. The number of subcascades per cascade is power-law distributed (Fig. 6).

Subcascade size. For each cascade the subcascades not only vary in number, but also in size. The distribution of the subcascade sizes taken across all the cascades is in Figure 6. Again, power-law fits describe the data well.

Subcascades are shallow. For a given node i and URL u, the distance from the subcascade root to i is an important metric characterizing the information flow in the network. Taken across all the subcascades, the maximum distance to root falls off exponentially (Fig. 7). The average distance falls off even faster.

The distances to the root are short, even when compared with the already short average path length in the follower graph (Fig. 2). One hypothesis explaining this data could be that when a user receives some interesting URL along an path longer than 1, then that user is very likely to start following the original source of the URL

[2]we parse out only the user name that immediately follows the `RT` characters

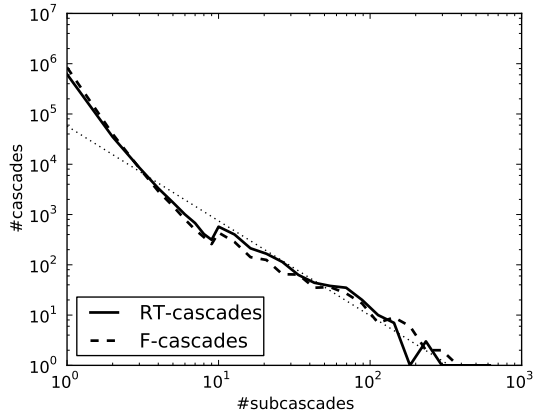

Figure 5: **Number of subcascades per cascade.** For each cascade we look at how many subcascades it consists of and plot the distribution. There is a clear power-law falloff (dotted line: $\propto x^{-1.83}$).

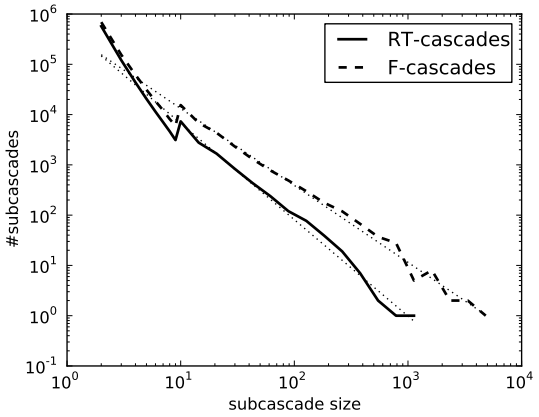

Figure 6: **Subcascade sizes.** The distribution of subcascades sizes taken across the set of all the subcascades. Power-law fits (dotted lines): $\propto x^{-1.9}$ for RT-cascades and $\propto x^{-1.5}$ for the F-cascades.

(subcascade root), thus shortening the potential future paths to one hop. Twitter does not place any constraints on the number of followers or followees, which allows the participants to optimize the information pathways for efficiency. Verifying this hypothesis would require additional data on how the follower graph evolves over time. We use the shallowness property of the cascades to reduce the computational effort of training our propagation model (§8), by considering only the one hop neighborhoods of nodes that have already tweeted a given URL.

The influence of the powerusers. A number of followers per user varies over several orders of magnitude (Fig. 1). The URLs tweeted by the highly connected users reach large audiences and are likely to be (re)tweeted by their followers. Figure 8 shows that indeed this is the case. However, the causality is likely to be bidirectional: the users's URLs are tweeted more because they have many folllowers, but also they have accumulated many followers because what they tweet tends to be interesting and viral.

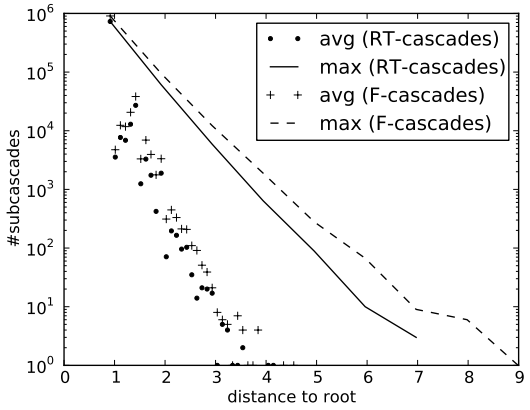

Figure 7: **Subcascades are shallow.** For each subcascade we measure the average and maximum distance from the root to each of the subcascade nodes. We plot the distributions of these values taken across all the subcascades. Both distributions fall off exponentially.

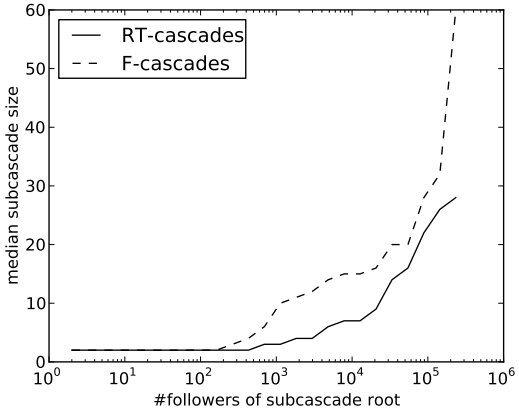

Figure 8: **The influence of the powerusers.** For all subcascades we plot the subcascade size against the number of followers of the subcascade root. For users that have more than approx. 500 followers there is a larger than 50% chance that if they are the root then their URL will be (re)tweeted more than once (i.e., median>1).

Rate of diffusion. When `@alice` tweets a URL some time elapses until her followers see that and tweet the URL. The *diffusion delay* for URL u and user i is the time from the moment the first of the followees of i tweeted u until the moment i tweeted u. The diffusion delay taken across all the (u, i) pairs is log-normally distributed with a median of 50 minutes (Fig. 9). The log-normal diffusion delay is incorporated into our propagation model (§7) to improve its accuracy.

7 Propagation model

Given the observations made in the previous sections, we construct two models of information propagation in social networks (§7.1). The models take into account the influence of users on one another, the virality of the URL and the diffusion delay. These factors have correspond-

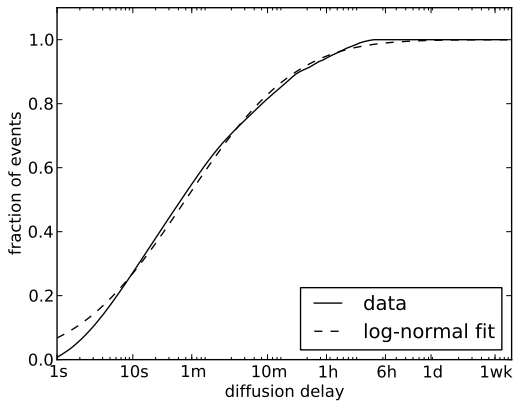

Figure 9: **Diffusion delay.** Log-normal distribution of the delay between the user first hears about a URL from one of the followees until the moment the user tweets about that URL. The fit is $\ln \mathcal{N}(\mu = 3.91, \sigma^2 = 6.86)$. The median is 50 minutes. All URL mentions, not only retweets are taken into account.

ing unknown parameters in the model. The optimal parameter values are found using the gradient ascent training algorithm (§7.2) with the goal of maximizing the number of predicted URL mentions while at the same time minimizing the number of false positives.

7.1 Tweeting probability

At-Least-One (ALO) model. Let $p_i^u = A(\alpha_{ji}, \beta_i, \gamma_u)T(\mu_i, \sigma_i^2, t_i^u)$ be the probability that the user i retweets URL u. The A component represents the time-independent (atemporal) part and T the time-dependent (temporal) part.

The A component is defined as:

$$A(\alpha_{ji}, \beta_i, \gamma_u) = 1-(1-\gamma_u\beta_i) \prod_{j:i\to j} (1-\gamma_u\alpha_{ji}p_j^u) \quad (1)$$

The parameters are as follows: $\gamma_u \in [0, 1]$ is the virality of URL u, $\beta_i \in [0, 1]$ is the baseline probability of user i tweeting any URL and the $\alpha_{ji} \in [0, 1]$ parameters define the influence of a followee j over user i. The notation $i \to j$ means "i follows j".

The expression for A is the probability that at least one of the events happens: 1) the URL u is viral (γ_u) and the user is influenced by some followee j (α_{ji}) that tweeted u with probability p_j^u or 2) u is viral and the user decides to tweet it by herself or is influenced by some unknown entity, as quantified by β_i.

The epxression for T follows our observations from §6.2 that the empirically observed diffusion time t_i^u of user i tweeting about u is log-normally distributed. The cumulative distribution function is parametrized by μ_i and σ_i:

$$T(\mu_i, \sigma_i^2, t_i^u) = \frac{1}{2} erfc(-\frac{\ln t_i^u - \mu_i}{\sigma_i\sqrt{2}}), \quad (2)$$

where *erfc* is the complementary error function.

Linear Threshold (LT) model. The ALO model assumes that it is enough to be influenced by one user to cause that user to tweet. The linear threshold (LT) model [6] generalizes over that by introducing a per-node threshold which must be exceeded by the cumulative influence from all the followees for the user to tweet. We replace the A component by:

$$A(\alpha_{ji}, \beta_i, \gamma_u) = s(\gamma_u(\beta_i + \sum_{j:i\to j} \gamma_u\alpha_{ji}p_j^u)), \quad (3)$$

where $s(x) = \frac{1}{1+e^{-a(b-x)}}$ is the sigmoid serving as a continuous thresholding function.

In our experiments, although the threshold is fixed, the α_{ji} parameters can be adjusted to achieve the effect of having a variable per-node threshold.

7.2 Training

The model is parametrized by $\alpha_{ji}, \beta_i, \gamma_u, \mu_i, \sigma_i$. These parameters are unknown and are learned in the following process. The training data set S consists of tuples $(i \in V, u \in U, t_i^u \in [0, \infty], F \in \{true, false\})$, where i is the tweeting user, u is the URL, F is the flag indicating whether u tweeted i and t_i^u is the *exposure time* measured as the longest time since one of the followees of i mentions u until either: 1) i mentions u ($F = true$) or if it does not then 2) t_{end}, the time at the end of the training data set ($F = false$). The positive events ($F = true$) cover all the actual URL mentions. The negative events ($F = false$) are generated as follows. For each positive event $(i, u, t_i^u, true)$ and each follower j of i that has not tweeted u, generate $(j, u, t_j^u, false)$. This reflects our empirycal observation that the URL cascades are shallow (§6.2). As we show in the evaluation (§8), including only one-hop follower neighborhood of the nodes that tweeted u provides sufficient training data for the model, while preserving the computational feasibility.

Optimization goal. Given the training data set S, the goal is to find the optimal set of parameters α_{ji}, β_i, γ_u, μ_i, σ_i. Optimality is defined as follows.

If the (i, u) pair occurs as an event in S, it does so only once either as a positive or negative event. If the event is positive then this defines the *target probability* $p_i^u = 1$ otherwise $p_i^u = 0$.

Our model defines the *estimated probability* $\hat{p_i^u} = A(\alpha_{ji}, \beta_i, \gamma_u)T(\mu_i, \sigma_i^2, t_i^u)$. The goal of the optimization is to bring the estimated probabilities $\hat{p_i^u}$ as close to the target probabilities p_i^u as possible.

The $\hat{p_i^u}$ variables are a function of other $\hat{p_j^u}$ variables. Finding the values of these variables would entail solving a system of non-linear equations, which is an intractable problem for the current $\hat{p_i^u}$ definitions. To overcome this,

we compute each $\hat{p_i^u}$ directly based on the known target probabilities p_j^u from the training data set (zeroes or ones).

Accuracy metrics. To measure the accuracy of the $\hat{p_i^u}$ predictions we borrow two concepts from the information retrieval literature: *precision* and *recall*. Let k_{tp} be the number of true positives, k_{tn} true negatives, k_{fp} false positives and k_{fn} false negatives. Then the precision $p = \frac{k_{tp}}{k_{tp}+k_{fp}}$ and recall $r = \frac{k_{tp}}{k_{tp}+k_{fn}}$. Intuitively, precision aswers the question: among all the positives predicted by the model, how many of them are true? and for recall: among all the positives, how many of them are predicted by the model?

The k_{tp}, k_{tn}, k_{fp} and k_{fn} numbers are obtained by comparing the true values p_i^u with the predictions of the model $\hat{p_i^u}$. In our case, p_i^u is binary, but $\hat{p_i^u}$ is a value from $[0, 1]$. We approximate the k values by computing their expected values given the estimated probabilities $\hat{p_i^u}$: $k_{tp} = \sum_{(u,i)\in S_p} p_i^u$, $k_{fp} = \sum_{(u,i)\in S_n} p_i^u$, $k_{fn} = \sum_{(u,i)\in S_p}(1 - p_i^u)$, $k_{tn} = \sum_{(u,i)\in S_n}(1 - p_i^u)$. The sums iterate over the trainig set S, where S_p is the set of positive events and S_n is the set of negative events. The precision p and recall r are then defined in terms of the k values as in the previous paragraph.

Ideally, we would like both the precision and recall to be high. A commonly used metric that combines the two is the *F-score*[3] defined as the harmonic mean of precision and recall $F = \frac{2pr}{p+r}$, which simplifies to $F = \frac{2k_{tp}}{2k_{tp}+k_{fp}+k_{fn}}$. The goal is to maximize F under the α_{ji}, β_i, γ_u, μ_i, σ_i parameters.

Iterations. We use the gradient ascent method for finding the optimal $F(X)$ under $X = (\alpha_{ji}, \beta_i, \gamma_u, \mu_i, \sigma_i)$. The process is iterative. Starting from the randomly initialized X_0 we obtain successively better approximations by computing $X_{t+1} = X_t + c\nabla F$, where ∇F is the gradient of F and c is the constant controlling the convergence rate. The parameters in X are forced to stay within their bounds in each iteration. After a fixed number of iterations we obtain the optimal value of X.

Computational complexity. Each iteration above computes a new value X_{t+1} as well as the gradient ∇F. Computing X_{t+1} involves 1) iterating through training set S and computing the $\hat{p_i^u}$ values, 2) computing the k values and 3) the F. In step 1) each α_{ji} is referenced at most once in the product and also each event in T is referenced at most once (one $\hat{p_i^u}$ computed per event), hence the computational complexity is $O(\max\{|S|, |X|\})$. In step 2) each event in T is reference at most once and step 3) has a constant cost. The complexity of computing X_{t+1} is $O(\max\{|S|, |X|\})$.

[3] http://en.wikipedia.org/wiki/F1_score

Computing ∇F involves computing the partial differentials of all the variables in X. This can be broken down to computing the differentials of individual terms comprising the sums in the k values. There are $|S|$ such terms. Each term can possibly reference only one γ_u among all us and only one α_{ji},μ_i,σ_i,β_i among all the is, hence computing ∇F involves summing over at most $5|S|$ terms (once for each of the 5 parameter classes α,β,γ,μ,σ). Computing each term involves iterating over the relevant α_{ji}, however each α_{ji} is referenced only once. The complexity of ∇F computation is hence $O(\max\{|S|, |X|\})$.

The iteration progresses until the desired numerical precision is reached. The computational complexity of m iterations is $O(m \max\{|S|, |X|\})$.

8 Model evaluation

8.1 Setup

Training and test data sets. We divide our 300 hour data set (§3) into two 150 hour parts. The *training data set* S is generated based on the events from the first 150h window excluding the URLs that were mentioned less than 5 times. For each URL u mentioned by user i we add a positive entry $(i, u, t_i^u, true)$ to S (§7.2). For each URL u not mentioned by user i, if there exists at least one followee j of i that mentioned u we add a negative entry $(i, u, t_i^u, false)$ to S. To reduce the size of the training data set, a negative entry is only added if i has tweeted some URL after j, otherwise we assume j's influence on i is too low to warrant an entry in the training data set. In other words, the models will only be trained to predict tweeting probabilities for the users that are one-hop away in the follower graph from the nodes that have mentioned u and only for arcs along which URL has been transmitted. Predicting mention probabilities for all (u, i) pairs (not only the one-hop neighborhood) is computationally challenging (§9).

For positive entries, the delay t_i^u is set to the time that elapsed from the beginning of the first 150h window till the first mention of u by i, but only if no followee of i mentioned u. Otherwise, t_i^u is set to the time elapsed from the earliest mention of u by a followee of i till the first mention of u by i (i.e., the diffusion delay, §6.2). For negative entries, the delay t_i^u is set to the time that elapsed from the earliest mention of u by a followee of i until the end of the first 150h window.

For the *test data set* we pick 100 random URLs that were mentioned at least 10 times both in each of the first and the second 150h windows. All the mentions of these URLs in the second 150h window become the test data set. The goal is to predict these events as accurately as possible.

There are approximately 700k positive and 9M negative entries in the training data set with 500k unique users and 50k unique URLs. The test data set has 5.2k URL mentions that need to be predicted.

Models. Our evaluation covers the following models (§7.1): **ALO** - the at-least-one model (§7.1), **LT** - the linear threshold model, **LTr** - a simplified version of LT that has one α_j parameter per influencing node instead of having an α_{ji} parameter for each pair of influencing and influenced nodes (this substantially reduces the computational effort). Finally, for the simple baseline, we use **RND**, a model making uniformly random decisions. The number of parameters in each model (the size of X, §7.2) are 12.5M for LT and ALO and 1.6M for LTr. The sigmoid parameters for LT are set to $a = 20$ and $b = 0.5$.

Training. We train all models starting from a random parameter initialization for 30 iterations of gradient ascent (§7.2). Training takes at most 20min on a single modern CPU core.

Metrics. Each model, after it has been trained, outputs p_i^u predictions in the range $[0, 1]$. We use the *discretization threshold* $\lambda = 0.5$ to classify the events into positive ($p_i^u > \lambda$) or negative ($p_i^u \leq \lambda$). For each u, we take the one-hop neighborhood of all the nodes that mentioned u. Within the scope of one-hop we compute the precision, recall and the F-score based on the discretized predictions and the true events from the test data set.

8.2 Results

Precision, recall, F-score. Figure 10 presents the precision, recall and F-score values for the one-hop neighborhood.

We observe the linear threshold model (LT) is able to correctly predict almost half of the URL mentions (55% recall) with at most 15% of false positives among the predictions (85% precision). It also gives the highest F-score out of all the models. Although negatives dominate the training and test data sets, LT learns the parameters affecting the positives well. The recall indicates that the model is unable to effectively learn the parameters for the 45% of positives in test set. These positives can most likely be explained by some other factors that the model does not account for. These factors are are most possibly external in nature as Twitter is a part of a much larger URL sharing ecosystem.

Controlling the precision-recall tradeoff: Some applications may require a higher precision from the model. The precision-recall tradeoff is controlled by varying the discretization threshold. The tradeoff for the LT model[4] is illustrated in Fig. 11. After the initial sharp increase in precision and decrease in recall, precision continues to increase without compromising on recall. Given that

[4]The curves for other models were similar

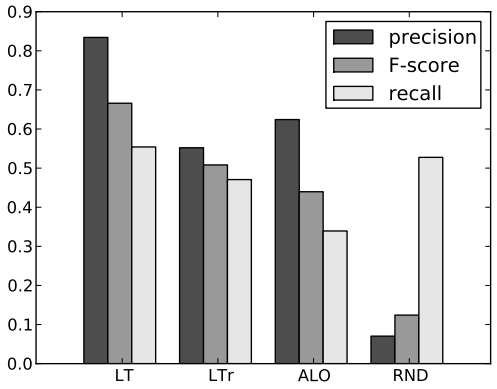

Figure 10: **Performance of the models.** For each model we measure the precision, recall and the F-score within the scope of the one-hop neighborhood of the nodes that have already mentioned the test URLs. The linear threshold model has the highest F-score and is able to predict more than half of the events (55% recall) at less than 15% false postive rate (85% precision).

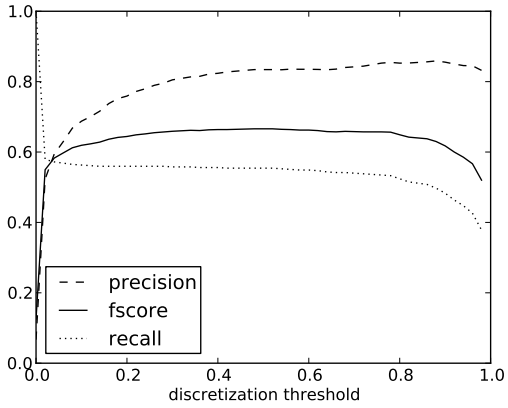

Figure 11: **The precision-recall tradeoff.** By adjusting the discretization threshold, the recall can be traded off for precision if the application requires it.

negatives dominate in our training data set, it underlines the ability of the model to learn the positives well and at higher thresholds it removes only the negatives, further improving precision. We are currently exploring alternative techniques for increasing the recall.

9 Discussion

The information propagation laws. In our 300 hour data set of 15M tweets we have observed several strong regularities. The user activity and the frequency of URL mentions are power-law distributed. The URL cascades are shallow with exponentially falling off height. They are composed of subcascades whose both number and size follow power-law distributions. We have also found that the diffusion delay follows the log-normal distribu-

tion extremely well.

Predicting the next hop and beyond. Our propagation model (§7) specifies the tweeting probabilities p_i^u as the function of the tweeting probabilities of the neighbors (p_j^us). However, during training that dependency is removed by substituting the neighbor p_j^us with the known values from the training data set. In this way, the set of p_i^us that need to be computed is determined by the training data set and the evaluation (§8) shows that if the training and prediction scope is narrowed down to the followers of the users that tweet URLs, our model predicts more than half of the events with less than 15% of false positives. Taking into account the empirical observation that most cascades are shallow (§6), the one hop neighborhood is likely to include most of the new URL mentions.

To be able to predict further than one hop away, the model would need to include p_i^us for more (u, i) pairs, however given the large number of URLs and the social network fan-out, this quickly becomes a computationally infeasible problem even for a two-hop neighborhood.

Another challenge is predicting not only the spread of the URLs but also the appearance of new root nodes initiating new subcascades. This requires a more detailed analysis of the conditions under which these events happen to be able to model them accurately.

Training. We have used the gradient ascent method for finding the set of model parameters under which the F-score is maximized. There is no guarantee that the global maximum has been reached. However, we have run training with different random seeds and arrived at models with identical (high) performance. Training could potentially be recast as another known combinatorial optimization problem, within which more precise optimality arguments could be made. This is currently under active investigation.

Continous model updates. Twitter is a real-time medium with a continuous stream of URL tweets. Keeping the model up-to-date while the new data is arriving is a separate and challenging problem. The iterative training algorithm we are using could be running constantly, following the changing optimal point in the parameter space. As new users and URLs appear, their corresponding parameters could be trained separately and more aggressively for faster alignment with the rest of the model. The training algorithm is embarrassingly parallel, composed mostly of large sums and can readily be scaled on frameworks such as e.g., MapReduce.

10 Conclusions

We have tracked the spread of 15M URLs over a 300 hour period on the Twitter microblogging service. Several statistical regularities are discovered: the power-laws in user activity, the exponentially falling off cascade depth, the decomposition of cascades into subcascades and finally the log-normal distribution of the diffusion delay. We also propose an information propagation model that predicts over half of the future URL mentions, while having only a 15% false postive rate. The model can serve as an important building block for several applications: personalized URL recommendation, filtering of incoming tweets and spam detection, problem areas we hope to explore further.

References

[1] ADAR, E., AND ADAMIC, L. Tracking information epidemics in blogspace. In *IEEE/ACM International Conference on Web Intelligence* (2005), pp. 207–214.

[2] BANERJEE, N., CHAKRABORTY, D., DASGUPTA, K., MITTAL, S., JOSHI, A., NAGAR, S., RAI, A., AND MADAN, S. User interests in social media sites: an exploration with micro-blogs. In *CIKM'09* (2009).

[3] CHA, M., MISLOVE, A., ADAMS, B., AND GUMMADI, K. P. Characterizing social cascades in flickr. In *WOSN '08*.

[4] DANAH BOYD, GOLDER, S., AND LOTAN, G. Tweet, Tweet, Retweet: Conversational Aspects of Retweeting on Twitter. In *Proceedings of HICSS-42* (2010).

[5] GOLDENBERG, J., LIBAI, B., AND MULLER, E. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing Letters 12*, 3 (2001), 211–223.

[6] GRANOVETTER, M. Threshold models of collective behavior. *American journal of sociology 83*, 6 (1978), 1420.

[7] HUBERMAN, B., ROMERO, D., AND WU, F. Social networks that matter: Twitter under the microscope. *First Monday 14*, 1-5 (2009).

[8] JACKSON, M., AND YARIV, L. Diffusion on social networks. In *Economie Publique* (2005), vol. 16, pp. 3–16.

[9] JAVA, A., SONG, X., FININ, T., AND TSENG, B. Why we twitter: understanding microblogging usage and communities. In *WebKDD/SNA-KDD '07* (2007).

[10] KEMPE, D., KLEINBERG, J., AND TARDOS, É. Maximizing the spread of influence through a social network. In *KDD'03* (2003).

[11] KRISHNAMURTHY, B., GILL, P., AND ARLITT, M. A few chirps about twitter. In *Proceedings of the first workshop on Online social networks* (2008), ACM, pp. 19–24.

[12] KWAK, H., LEE, C., PARK, H., AND MOON, S. What is Twitter, a social network or a news media? In *WWW'10*.

[13] LERMAN, K., AND GALSTYAN, A. Analysis of social voting patterns on digg. In *Proceedings of the first workshop on Online social networks* (2008), ACM, pp. 7–12.

[14] LESKOVEC, J., MCGLOHON, M., FALOUTSOS, C., GLANCE, N., AND HURST, M. Cascading behavior in large blog graphs, 2007.

[15] NEKOVEE, M., MORENO, Y., BIANCONI, G., AND MARSILI, M. Theory of rumour spreading in complex social networks. In *Physica A: Statistical Mechanics and its Applications* (2007), vol. 374, pp. 457–470.

[16] SZABO, G., AND HUBERMAN, B. Predicting the popularity of online content. *Communications of the ACM* (2008).

[17] WU, F., HUBERMAN, B., ADAMIC, L., AND TYLER, J. Information flow in social groups. *Physica A: Statistical Mechanics and its Applications 337*, 1-2 (2004), 327–335.