Revisiting Models of Influence Spread in Social Networks

Mitchell McIntire
Department of Computer Science
Stanford University

***Abstract*—In many settings, network structure is used to inform difficult decisions, such as how promotional events should be targeted to encourage widespread adoption of a product, or predict complex phenomena, such as how contagions will spread through social circles to create an epidemic. Though seemingly unrelated, modeling these phenomena using networks reveals that they are strikingly similar.**

Research on influence maximization aims to capture these scenarios within a general problem setting: given a network, find a set of network nodes to 'activate' in order to encourage a maximal spread of network activity (e.g. adoption of a product). However, this research is typically done under strict modeling assumptions which are at best intuitive but unjustified. The Linear Threshold Model introduced by [1] is one example of this. The model's simplicity is necessary to some degree in order for the influence maximization problem to be manageable (as proved in [1]), but the model is never justified, nor are the algorithms developed under this framework ever evaluated on data that is not synthetically generated by the theoretical model.

Our contributions in this project are threefold. First, we compile a very large data set (approximately 70 GB) into real-world influence spread events, a type of data which is extremely scarce in practice. We then extensively analyze common spreading models against this data set, which is to our knowledge the first time these spreading models have been evaluated for their usefulness in describing real spreading phenomena. Finally, we propose an alternative model of influence spread phenomena, and show that it models our data substantially better than the widely studied Linear Threshold Model.

I. Introduction

We begin by giving an overview of the intuition and motivation for the influence maximization problem. Suppose we are on the PR staff of a company which sells a product popular with teenagers. Our goal is to maximize sales, and to do so we have a promotional budget in the form of discount coupons or free samples. How should we distribute this budget? Clearly, we suspect that 'popular' teenagers might sway their peers to adopt our product if they are using it. Domingos and Richardson [2] suggest using social networks to answer this question - intuitively, 'popular' nodes may inspire their peers in general settings much like they will in our simple (at least, simply stated) marketing problem. As we will see, we can do better than just using heuristics such as node degree to guide this process.

More broadly, we can state this problem as follows: given a directed graph $G = (V, E)$ and a budget k, select a subset $S \subset V$ with $|S| \leq k$ to 'seed' with initial influence. The objective is to maximize the expected number of nodes which become infected, denoted by $\sigma(S)$. To approach this problem rigorously, we must formalize a notion of how influence spreads, since without this we cannot reason about expectations over the spreading process. We will roughly follow the approach of [1], who first modeled this problem on a node-by-node basis and framed it as a discrete optimization problem. Most subsequent research on influence maximization have used the same or very closely related models. We will refer to these models as 'spreading models' or 'spreading processes'.

II. Related Work

We'll first discuss the results given in [1]. The key insight of this work was that some fairly natural choices for spreading models lead to objective functions σ which are submodular, i.e. which follow the rule that $\sigma(S \cup \{v\}) - \sigma(S) \geq \sigma(T \cup \{v\}) - \sigma(T)$ for all v whenever $S \subset T$. Intuitively, this result says that adding new nodes cannot suddenly become more impactful later in the selection process. This leads to an important result: it is known that greedy algorithms work well in optimizing submodular functions, and therefore we expect greedy algorithms to perform well in influence maximization. In particular, despite this problem being NP-hard, the greedy hill-climbing approach of choosing the maximal node at each step achieves at least a $(1 - 1/e)$-fraction of the optimal performance. In other words, we first choose a $v \in V$ which maximizes $\sigma(\{v\})$, then iteratively find nodes that increase σ by the largest amount possible. This leads to a simple algorithm for influence maximization (given a method of computing σ).

We'll revisit this result, but first we should backtrack and discuss the framework in which these results were proved. Namely, we need to formally define a spreading model. Kempe et al. [1] discuss two main models: the Linear Threshold Model and the Independent Cascade Model. In the Linear Threshold Model, each edge is given a directional weight, i.e. for each $u, v \in S$ we have a weight $b_{u,v} \in [0, 1]$. Furthermore, weights are normalized so that $\sum b_{u,\cdot} \leq 1$, where the sum is over neighbors of node u. Then each node $v \in S$ picks a threshold $\theta_v \leftarrow [0, 1]$ uniformly at random, and the process begins iterating. In each step, a node u becomes active if the sum of the weights to its active neighbors exceeds its threshold, i.e. if $\sum_v 1\{v \text{ is active}\} \cdot b_{u,v} \geq \theta_u$. Here $1\{\cdot\}$ is the indicator function, and we assume $b_{u,v}$ is zero if u and v are not connected.

Our influence function $\sigma(S)$ is then defined as the expected number of active nodes at the conclusion of this spreading procedure[1], with nodes in S initially active and other nodes

[1]Note that this expectation is with respect to the randomness in threshold assignments.

inactive. The alternative model of [1], the Independent Cascade Model, is similar, but we won't explicitly discuss it here for the sake of brevity. It is of a similar flavor to the Linear Threshold Model, and the results we'll discuss generally apply to both models [1]. These models have in some sense been unified into the more general triggering model, which is also discussed in [1].

Given this spreading model, Kempe et al. run the greedy algorithm by, in each iteration, simulating the random assignment of thresholds, computing spreads, and averaging over, say, 10000 trials to determine the marginal benefit of adding each node to the current seed set. However, running these full simulations are time consuming, and they must be done for every node, leading to an $O(kn)$ running time with a substantial constant factor. Following work has aimed to address this inefficiency, e.g. in [3], [4]. As it was proved in [1] that the greedy algorithm achieves the best possible approximation factor, these methods have the goal of matching greedy performance without incurring the same computational cost.

In [3], it is explicitly proven that computing σ exactly in polynomial time is impossible unless $P = NP$. Their insight is that influence computation on directed acyclic graphs (DAGs) can be done much more efficiently, and that influence can be mostly thought of on a local scale. However, their simplification in terms of local DAGs still results in a computational problem which is NP-hard, leading them to use a heuristic subroutine to approximately find good local DAGs, which has no approximation guarantees with respect to the main local DAG construction objective.

This approach is intuitively somewhat unsatisfying: we are approximating our true objective, influence maximization, with the greedy hill-climbing algorithm. The greedy algorithm is not efficiently computable, so a local DAG subproblem is introduced with the goal of approximating influence computations in the execution of the greedy algorithm. Then, we must use another heuristic approach to approximate the solution to the local DAG subproblem, this time without approximation guarantees. All of these approximations and heuristics layered together seem like they must surely have compounding errors. We'll discuss these 'layers' of complexity, and how tightly coupled algorithms like this are to the particular mechanisms of the particular spreading model, in more detail below (specifically, we'll draw analogies to overfitting in machine learning, and posit that some of the performance gains of these algorithms do not reflect a better understanding of real-world spread phenomena).

The local DAG approach is still promising, however, as [3] demonstrates that it is about three orders of magnitude faster than the Monte-Carlo based procedure of [1], and provides better influence spread results than methods of comparable scaling on some datasets (and more consistent results in general, though the inconsistency of the other methods is not explored or explained). It's also worth noting here that the authors of [3] explicitly attribute their results to "the design of LDAG tailored specifically to the LT influence model", where LT here stands for linear threshold. This is indeed a natural influence model, but neither paper has even addressed the possibility that the model does not fit real-world influence spread phenomena. All experiments thus far have been carried out on real networks, but with a synthetic 'contagion' whose spread is dictated only by the linear threshold spreading model. We will discuss this much more below.

In [4], a different approach is taken. Rather than using subproblems which intuitively should give good results, [4] introduces several optimizations and pruning strategies to augment the more naive greedy approach. The details of many of these optimizations are difficult to summarize concisely, so for now I will defer them. However, it is demonstrated that the resulting algorithm, called SIMPATH, has faster runtime than LDAG by a significant margin, uses dramatically less memory, and has comparable or somewhat improved influence spread performance.

An interesting approach was taken by Goyal et al. in [5], in which the spreading models we've already discussed are used but the edge influence weights (which previous works have set to be uniform or random) are estimated directly from data. Various heuristics are compared for how exactly these influence weights should be computed, but the idea is to use an *action log* which contains a list of actions for each user and time stamps for those actions. This provides a way to infer dependencies between users actions, and possibly informs how influence weights should be set. The authors also consider the related problem of predicting *when* a node will become active under a continuous-time modeling of the network. This prediction problem, together with their continuous-time model which allows for this reasoning, is an interesting departure from the typical approach of the other papers cited here, which use discrete time steps $t \in \{0, 1, 2, \ldots\}$.

III. SPREADING MODELS AND EVALUATION

In this section, we begin by briefly reviewing the Linear Threshold and Independent Cascade spreading models, then discuss how we can evaluate spreading models for their agreement with real data. Note that we primarily restrict our attention to the Linear Threshold Model for now, since a large portion of influence maximization research has focused on this model and since the conventions with respect to its parameterization are easier to manage in our setting.

A. The Linear Threshold Model

To review, we are given a directed graph $G = (V, E)$ in which an edge $(u, v) \in E$ indicates that it is possible for node u to influence node v. Furthermore, each edge $(u, v) \in E$ has a weight $b_{u,v} \in [0, 1]$, with the sum of incoming edge weights at most one for every node. This weight intuitively represents the extent to which node v is influenced by node u. Each node $v \in V$ is assumed to choose a threshold θ_v uniformly at random in $[0, 1]$. At a given time, each node is either active or inactive, and an inactive node v becomes active in the next time step if $\sum_{u \in U} b_{u,v} \geq \theta_v$, where this sum is over the set U of active parents of v.

Given a particular assignment of node thresholds θ_v, and given the edge weights $b_{u,v}$, the influence spread estimation becomes straightforward, as the spread can be directly simulated beginning with the active seed set $S \subset V$. The influence maximization problem in the Linear Threshold Model assumes that the edge weights are given, but that thresholds are randomly selected, and so to determine the spread of influence, various

Monte-Carlo or other methods are used (as we've discussed above).

In this project, we make the fairly standard assumptions that edge weights are uniform and for each node sum to one. To be precise, for each node $v \in V$ we have $b_{u,v} = 1/d$ for each edge $(u, v) \in E$, where d is the in-degree of v. The alternative to this which is sometimes used instead is to generate edge weights randomly, but it's unclear that there is any good reason to do this. The most compelling method of setting the edge weights is the empirical approach of [5], [4], in which edge weights are set based on activity co-occurrence between pairs of nodes, but this approach leads to scaling problems for the dataset used here (which are surmountable, but perhaps not within the time frame of this project).

B. Evaluating the Linear Threshold Model

Suppose we are given a dataset which contains entries of the form (G, L), where $G = (V, E)$ is a directed graph with edge weights defined as above and L is an activity log containing pairs $(t, v) \in \mathbb{N} \times V$ indicating that node v becomes active at time t (we may as well assume that nodes which become active at $t = 0$ are seed nodes, and that each node v appears in the log L at most once, at the earliest time at which v participates).

To evaluate the model, we treat the dataset as a sample from the ground truth distribution of real influence spread processes, and use a log-likelihood analysis to evaluate how well the model fits real data. This analysis can be used to guide not only model selection but also parameter selection within models, for example to decide whether edge weight selection schemes make a model more realistic.

To compute the log-likelihood of a dataset $\{(G_i, L_i)\}$, we want to compute an average over each 'point' in the dataset. This reduces the problem to computing a log-likelihood for a single (G, L) pair. To do this, we make the observation that, since in the Linear Threshold Model each node chooses its threshold independently and uniformly at random, we can reason about each node somewhat separately. Formally, we have that the activation of a node is conditionally independent of the rest of the graph given the activity (or inactivity) of its parent nodes.

This allows us to compute a node's individual likelihood somewhat simply: we lower bound the node's threshold by recognizing that it did not become active earlier, and upper bound its threshold by noting that if its threshold were higher, it would not have become active when it did. More precisely, suppose node v becomes active at time t'. Then at time $t' - 1$ and all times previously, the amount of influence which v was subjected to must have been insufficient for v to become active. Denote by $I_v(t)$ the influence node v is subjected to at time t (noting that $I_v(t)$ monotonically increases in t). Then if node v becomes active at time t', it must be that $\theta_v \in (I_v(t' - 1), I_v(t')]$, since if it were lower, it would have activated before time t', and if it were higher, the influence at t' would be insufficient to active it. Similarly, if v never becomes active we can lower bound the possible values of θ_v by the maximum influence v is subjected to.

For node v this yields a threshold interval $\Theta_v \subset [0, 1]$ in which θ_v must lie for the observed behavior to be possible. Then $|\Theta_v|$, the size of the interval, represents the node likelihood, and the log-likelihood is computable as $\sum_{v \in V} \log |\Theta_v|$. Note, however, that since we are averaging over graphs of different sizes, our result will be more heavily weighted by larger graphs. To remedy this, we can also normalize this result by the number of nodes $|V|$ in G, but this is not strictly necessary (in some sense, by not normalizing we are instead looking at individual nodes' activation patterns as data points, rather than entire spread events).

Note that under the Linear Threshold Model (with normalized weights), a node which has all of its parents active must activate, but in practice it is not uncommon to see this fail. For example, a user might only follow one person, but still fail to re-tweet something that that person tweets. This is not particularly surprising, and should probably not lead us to assign the event a 0% probability under the model, so we employ smoothing: for nodes with in-degree d and which have zero likelihood under the Linear Threshold Model, we assign a smoothed likelihood of $1/(d + 1)$. This intuitively assigns a moderately high likelihood to nodes with fewer parents, while for example if a user follows dozens of users and all of them re-tweet something, that node is assigned a very low likelihood if it does not re-tweet it as well.

C. The Independent Cascade Model

In the Independent Cascade (IC) model, we are again given a directed graph with weighted edges. The primary difference in this model when compared to the LT model is that edge weights are now given in $[0, 1]$, and correspond to probabilities of child nodes becoming active. More precisely, in the IC model, when a node u becomes active, for every edge $(u, v) \in E$ the node v becomes active (if it was not already) with probability $w(u, v)$. Thus each edge is considered with respect to the influence spread at most once.

A typically treatment of the IC model might estimate the edge weights using the data, for example by looking at tweet co-occurrence between users that follow each other. However, this approach is beyond our scope for this quarter, as it requires nontrivial thought, implementation, and computation to calculate and test weights. We instead take the approach of each node v having an activation parameter $\theta_v \in [0, 1]$, and assign the weight of an edge (u, v) to be $w(u, v) = \theta_v$. In other words, we make the simplifying assumption that a node is influenced equally by all of its parents.

This assumption is obviously not justified in many settings, but it reduces the IC model to a case which is much more tractable. For example, if we assume that the node parameters θ_v are chosen uniformly at random from $[0, 1]$, then we can compute the probability that a node became active after k of its parents became active as $(1 - \theta_v)^{k-1}\theta_v$. This is just the geometric distribution, and we can easily integrate over the possible values of θ_v to get the node probability $1/(k^2 + k)$. Similarly, we can compute that the probability a node does not become active given k active parents is $1/(k + 1)$.

Here we arrive at a problem - from the above, we can see that all node behavior can be summarized simply in terms of these quantities, and that this yields that all node behavior becomes very unlikely given more parent activity. Intuitively this is because if many parents become active, the

node ought to have become active sooner for most values of θ_v. Furthermore, if the node eventually does activate, this behavior is even more unlikely, because θ_v is probably low (if k is high), but the node still became active (which is unlikely given a low θ_v).

The above is a problem because in some sense we never look at nodes favorably. In the LT model, a node with several parents which does not become active given only one active parent is given a high likelihood ($1 - 1/d$ for in-degree d). The IC model by comparison yields very low likelihoods as a result, and so we treat it relatively briefly here (and do not include our experimental results for this model, though our code includes a full implementation).[2]

D. The Sigmoid Threshold Model

In this section, we describe our proposed model of influence spread. This model is a generalization of the LT model, but not to the extent of the general threshold model discussed in [1]. This general threshold model of [1] still incorporates node thresholds, but now determines whether a node v is activated by checking the inequality $f_v(S) \geq \theta_v$. Here θ_v is the node threshold, and f_v is a general function of the set S of active parents of v. This model was proved inapproximable in [1].

In this project, we introduce the Sigmoid Threshold (ST) model by defining $f_v(S) = \sigma(w \cdot |S| + b)$, where $\sigma(x) = (1 + e^{-x})^{-1}$ is the sigmoid function (we will no longer use σ to denote the influence function, as we did in Section II). Here, we introduce global parameters w and b which scale the input appropriately. These parameters will be tuned to maximize the log-likelihood of the data (they are global in the sense that they hold for the entire data set, and are not maximized separately for each node or for each separate spread event). For the remainder of this section we assume that w and b are fixed, and that $\sigma(x)$ is the sigmoid function evaluated at $wx + b$.

Evaluating the log-likelihood of our data under the ST model is then similar to the procedure for the LT model: if a node v does not become active with an active parent set S_0 but does for an active parent set S, then it must have $\theta_v \in (\sigma(|S_0|), \sigma(|S|)]$. The size of this interval corresponds exactly to the likelihood assigned to this node behavior. Similarly, if a node never becomes active, its threshold must lie in $(\sigma(|S|), 1]$, and we can compute its likelihood directly.

A key observation here is that node in-degree does not influence these likelihood computations. This is significant, and we'll discuss it in much more detail later. We do admit one use of node in-degree, in the case discussed at the end of Section III-B: if a node with in-degree d is assigned zero probability due to peculiarities in the data, it is instead given a smoothed likelihood of $1/(d + 1)$. This is justifiable in this setting for the same reasons as for the LT model, and will not affect comparison between these models.

We note that the ST model as introduced here does not yield a submodular influence function. This is significant, as all of the progress made (to my knowledge) on the influence maximization problem is built upon submodular influence functions. This is discussed above, but to reiterate, the influence maximization problem under most nontrivial spreading models is NP-hard. The submodularity of the influence function enables a $(1 - 1/e)$-approximation based on greedily choosing seed nodes.

To see that the ST model influence function is not submodular, consider the simple network with many nodes $\{u_1, u_2, \ldots\}$ and a single node v. Each u_i has in-degree zero and out-degree one, with a single edge pointing to v. Consider that, over random assignment of θ_v, the probability that v becomes active when k of the u_i are active is exactly $\sigma(k)$. But note that the incremental probability from adding additional u_i is $\sigma(k+1) - \sigma(k)$, since θ_v was drawn uniformly from $[0, 1]$. It is easy to see that for moderate k this expression is larger than for very small k, and thus submodularity fails.

The above does highlight a way in which we can modify the model so that it yields a submodular influence function. In particular, we need to choose node thresholds from a distribution with CDF F such that $F(\sigma(k + 1)) - F(\sigma(k))$ is non-increasing. One possibility is to choose F so that the above difference is constant, i.e. to choose thresholds from the uniform distribution on the image of $\sigma(\cdot)$. It is not too hard to show that this is sufficient for the resulting influence function to be submodular (we omit the proof here for space). We did not have time to explore this within the scope of our project, but this could lead to an interesting distribution over node thresholds, and allow a straightforward extension of the existing research on influence maximization to our model.

IV. Data Collection and Preparation

Here we describe the data used in our experiments, and outline how this data was processed into spreading events. Our data has two main components: user tweet data from the social media website Twitter, and a user follower graph which shows roughly which users are exposed to others' tweet content.

A. The Twitter7 Dataset

The Twitter7 dataset [6] consists of approximately 480 million tweets from between June and December of 2009. This data contains for each tweet the user handle of the poster, the time at which the tweet was posted, and the text-based content of the tweet. These tweets were generated by about 17 million unique users. For our purposes, we are particularly interested in *re-tweets*, which occur when a user shares a tweet by another user to their own followers. We find that there are approximately 72 million re-tweets within this dataset.

To begin processing this data, we first identify re-tweet content in general, i.e. we filter the 480 million tweets and find the content that was re-tweeted at some point. This reduces our scope to about 72 million tweets, which we further reduce by requiring that the content is re-tweeted at least 100 times to be considered a re-tweet event. We also ensure that the text content of the tweet is at least 13 characters. In most cases, tweets which are widely re-tweeted without meeting this requirement appear to be images, with little or no captioning. This constraint also removes instances of highly common re-tweet content, such as "...", which is widely re-tweeted but

[2] We also considered the case where node parameters are chosen to maximize likelihood. This case is tractable as well, requiring closed-form maximization instead of integration, but results in setting all inactive node parameters to zero. This yields very high likelihood, but is hardly reasonable.

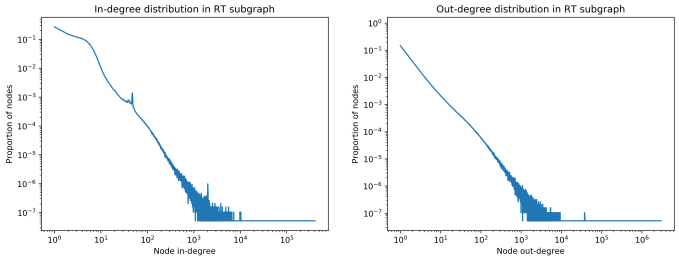

Fig. 1: In- and out-degree distribution for subgraph of re-tweet event participants

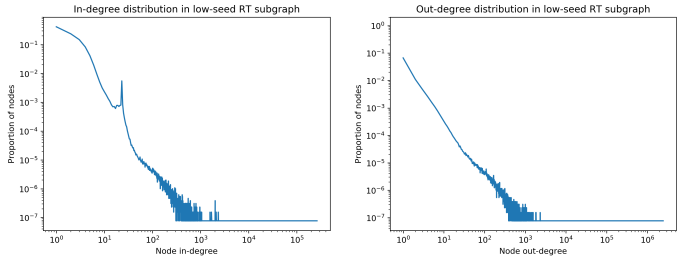

Fig. 2: In- and out-degree distribution for subgraph of re-tweet event participants used in our experiments

likely captions shared images or links, and is not necessarily restricted to a single tweet event. Since users can and often do re-tweet the same content multiple times (e.g. when there is a giveaway for which re-tweets constitute entries), we further reduce our focus to the first instance in which users participate in a given re-tweet event. These restrictions collectively leave us with approximately 500,000 tweets spread across 1,376 re-tweet events.

B. The 2010 Twitter Follower Network

To make use of the above tweets, we need a network that captures the relationships between users. Because the tweets in our dataset occur up until late December of 2009, we use a Twitter network snapshot from 2010 to have as many users as possible in the network (since a snapshot from 2009 may exclude a large portion of people tweeting in late 2009, especially given that Twitter was fairly young at the time).

The Twitter network we use [7] contains about 41 million nodes and 1.5 billion edges. Each node is a distinct user, while an edge from user i to user j denotes that user j follows user i and therefore observes their tweet content. The edge list encoding of this network is over 25 GB and is entirely unmanageable within my computational budget, and so the first step in using this data was to project this graph to the subgraph on only user nodes which participated in the re-tweet events identified above. This reduced the size of the user graph by approximately 90%, yielding a network with 19 million nodes and 139 million edges. Note that the number of nodes was reduced proportionally much less than the number of edges.

The degree distribution of the re-tweet subgraph is shown in Figure 1. We do not generate degree distribution plots of the full Twitter network for computational reasons, but these distributions are available online.[3] Note that both the in-degree and out-degree distributions of our subgraph are very close to that of the full graph.

Unfortunately, our data is not perfect, and in many cases we have problems with overlaying the re-tweet event onto the Twitter user network. In rare cases, re-tweeters are not in our network at all. Often, the parents of re-tweeters (which connect them to the re-tweet event) do not have their tweet recorded in the Twitter7 dataset. In general, we often find that users re-tweet content despite none of their parents in the subgraph apparently having tweeted that content themselves. In these cases, we assign these users as seeds for the event (with a delayed start time, since typically it is assumed that all seeds are active at $t = 0$).

This problem is unfortunately relatively prevalent in our data, and as a result we find that most of our re-tweet events consist almost entirely of 'seed' users when using this solution. We therefore further prune re-tweet events to consider only those in which at most 25% of the participating users are seeds. This leaves us with 87 re-tweet events, and further reduces the size of the subgraph under consideration to 13 million nodes and 36 million edges. Note that this does not mean that there are 13 million unique re-tweet participants in these 87 events; we include all parents and children of the participants as well, since these are relevant to the spreading dynamics. The degree distribution of this subgraph is shown in Figure 2. The distribution is similar to that of the larger subgraph, but the higher-degree nodes have largely been removed.

V. Experiments

Our experiments are focused primarily on evaluating spreading models for their agreement with real data. The key departure from the discussion above that we must make is to specify what amount of time is 'significant.' In particular, suppose that a node becomes active a few seconds after one of its parents. This might not indicate that the node was influenced by that parent if it already was influenced by other nodes for a longer period of time. Fortunately we can approach this using a single parameter Δ. Intuitively, Δ is the amount of time we assume elapses after a node becomes active before its children are necessarily influenced by that activity.

A. Choosing the Δ parameter

Our first analysis is of the log-likelihood of our data under the Linear Threshold (LT) model. We can compute this as described in Section III-B, and vary Δ to see how this quantity changes as node attention is assumed to be more or less prompt. This relationship is shown in Figure 3. As discussed previously, we can see that the data log-likelihood is monotonically increasing with Δ, since as Δ goes to zero the viable window for the active node thresholds becomes just $1/d$ for nodes with in-degree d.

Since the relative change in log-likelihood over Δ is extremely small for Δ at least about one minute, we will forgo further discussion of the effect of our choice of parameter Δ. The rest of our experiments which require the Δ parameter

[3] http://law.di.unimi.it/webdata/twitter-2010/

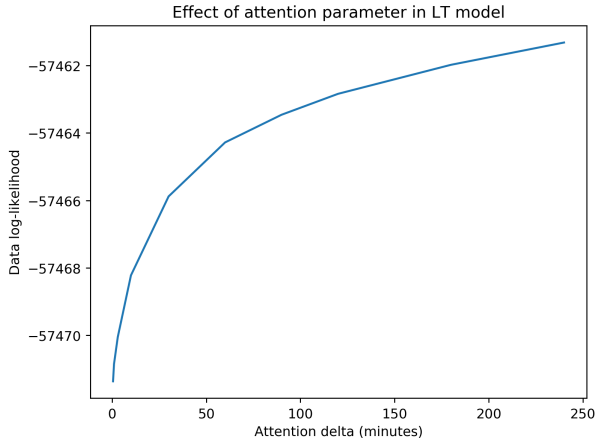

Fig. 3: Average log-likelihood of Twitter RT events as Δ, the time before users are assumed to have observed re-tweeted content, is varied

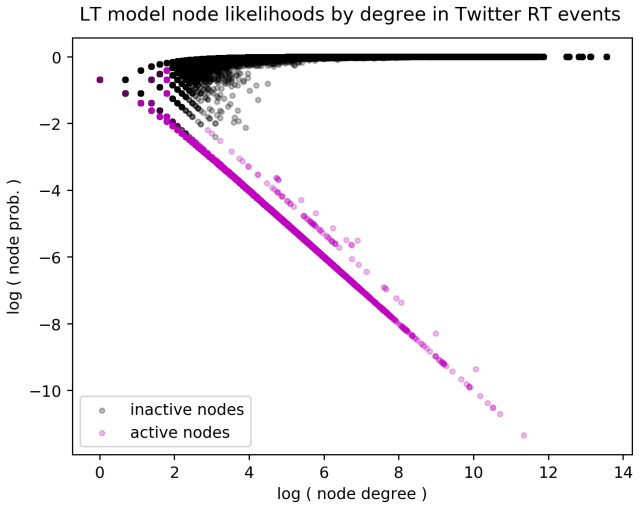

Fig. 4: Log-likelihood of nodes by degree in Twitter RT spread events

(which are any that calculate log-likelihoods of real data in the LT model) are done with a fixed Δ of three minutes.

B. Evaluating data likelihood under the LT model

Our next set of experiments are the first of the main results in this project. In particular, we carefully evaluate the log-likelihood of real-world influence spread events under the LT model, and analyze the 'problem areas', i.e. the spread patterns in the data that consistently yield a low likelihood under the LT model.

To do this, we generate plots of each spread event, with nodes colored by the likelihood that they are assigned by our evaluation under the LT model. An example of this is shown in Figure 6. Note that while this example is cherry-picked, this cherry-picking is only done with respect to the readability of the plot - in particular all of the process dynamics which

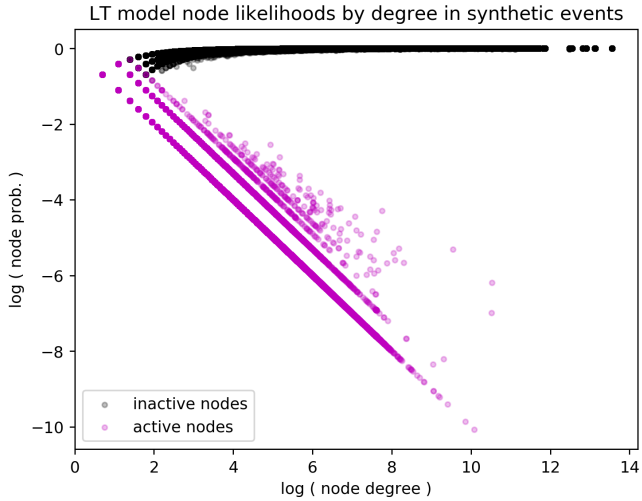

Fig. 5: Log-likelihood of nodes by degree in synthetic LT events

we discuss are highly typical of all of our events (besides one particularly unusual event, which I'll include below if there is space).

Observing the event evolution depicted in Figure 6, we can see as early as just 30 active nodes into the process that nodes with high in-degree tend to have lower likelihoods (denoted by darker coloring). Note that the pure white nodes at the top of the plot are seed nodes, and that only active nodes are shown (because the set of inactive nodes which are exposed to the event is far too large to plot). The effect of high in-degree on node likelihood behavior is interesting; for inactive nodes, high in-degree is generally favorable, since these nodes follow many users which are not involved in the RT event and therefore might have a threshold anywhere in, say, $(c/d, 1]$ if c of its parents re-tweeted the content.

By contrast, in our plot of active nodes, we can see that nodes with high in-degree receive extremely low likelihood scores for becoming active. This pattern exists across our entire data set, and is intuitively problematic: users with high in-degree are exactly the users we expect to participate in spread events, and the Linear Threshold model's framework explicitly limits the probability of node participation if a large portion of those it follows are outside the event.

We examine this phenomenon from another perspective in Figure 4. Here, we compile all users across our entire data set, and plot them by the likelihood of their behavior in the event(s) in which they participate and their in-degree (all points correspond to users which had at least one active parent in a given event; seed nodes are not shown). Note the striking difference in assigned likelihood based on whether the users re-tweet the content or not. For users which do not become active high in-degree yields high likelihood. The slight curve visible for inactive nodes in the upper-left of Figure 4 exactly corresponds to the line $1 - 1/d$ for in-degree d, which is the upper bound on possible node likelihood.

For active nodes, the distribution of node likelihood is very

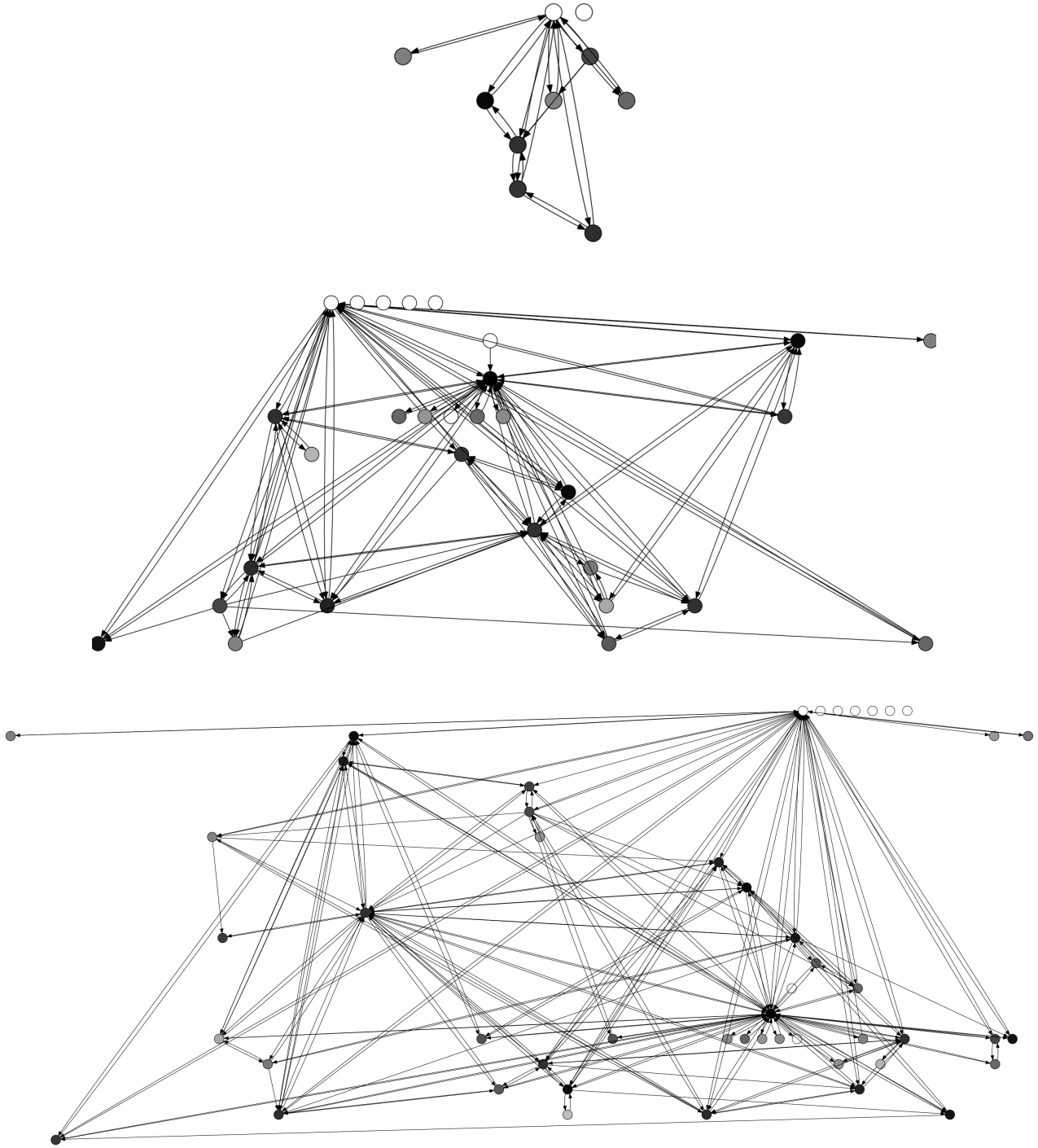

Fig. 6: Spreading process for a particular re-tweet event regarding a giveaway for a video game console. Darker nodes indicate lower likelihood of the observed behavior. The figures respectively indicate the first 10, 30, and 50 users who re-tweeted the content.

different. We observe an inverse linear relationship between node in-degree and likelihood due to the tendency of high-degree nodes to become active after relatively few of their parents become active. It is interesting to note that these bizarrely different relationships between in-degree and likelihood for active and inactive nodes both stem from the same cause: nodes, especially high-degree nodes, tend to have a vast majority of their parent nodes located outside the event graph (i.e. many of their parents are never exposed to the event).

More intuitively, users who follow a huge number of accounts on Twitter probably roughly partition the users they follow according to their interests; e.g. they follow professional athletes, musicians, and their family. Re-tweet events within one of these circles are unlikely to reach other circles. Because of this, under the LT model the user is deemed less likely to re-tweet content from the musicians they follow because of their activity in other communities. In reality, we would not expect this to necessarily be the case: a user who is active in many circles might even be more likely to participate in some events. In this sense, the Linear Threshold model fails to properly model the spreading process.

Another method we can use to examine the LT model against our data is to generate synthetic spreading events according to the LT model. We summarize this experiment in Figure 5. As expected, the likelihood distribution looks similar to the real data, since it is computed in the same way. However, it is worth noting that there are substantial differences between the distributions, both for active and inactive nodes. For inactive nodes, real data has a much larger fraction of lower-likelihood nodes, specifically in the range between .01 and .1 node probability. Recalling that both event types have identical degree distribution (since they are on the same network), we can also see that there are many more high-likelihood active nodes in the synthetic spreads as well.

This indicates that synthetic events are given much higher likelihood than real events by the LT model. More significantly, it shows a distinct difference in distribution between the real and synthetic data. This is noteworthy because all prior work in this area to our knowledge assumes the LT model or something similar, and is therefore addressing an event distribution which differs fundamentally from real events. It is this difference which motivates our proposed model.

C. Evaluating the Sigmoid Threshold Model

To evaluate the ST model, we first must fully define it. In particular, we still need to choose the sigmoid parameters w and b. Unfortunately, choosing these parameters to maximize the log-likelihood of our entire data set is not only analytically intractable but computationally infeasible as well due to the size of our data set. We instead select w and b by maximizing the log-likelihood of each event separately using L-BFGS-B [8], and analyzing the distribution of the parameters for each event. This approach leads us to set $w = 0.35$ and $b = -10$, yielding a threshold function $f_v = \sigma(0.35 \cdot |S_v| - 10)$ for node v with active parents S_v. This function is shown in Figure 7.

We then repeat our evaluation of log-likelihood of our data set on the ST model. Under the ST model the likelihood distribution over node degree is no longer as interesting, so we restrict our attention to comparing likelihoods of the data

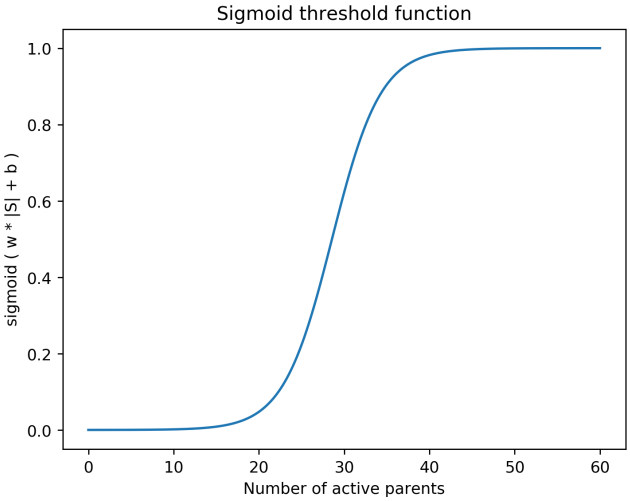

Fig. 7: Our sigmoid threshold function

under the ST and LT models. Our results here are shown in Figures 8 and 9.

In Figure 8, we demonstrate that while event log-likelihoods are correlated between the two models, the ST model assigns substantially higher likelihoods to the data than does the LT model. It is interesting to note that there is exactly one of our 87 events that is assigned a higher likelihood under the LT model than under the ST model, and it is exactly the 'unusual' event that we referred to near the beginning of Section V-B. This event is shown in Figure 10. Given the text of the tweet, it appears that this event was the product of coordinated accounts, perhaps bots rather than real users. That this event is assigned a higher probability by the LT model might be an additional endorsement of the ST model's better agreement with real data, but this example is strange enough that we hesitate to declare this conclusively.

Finally, we observe Figure 9 for a more thorough comparison of node probabilities under the respective spreading models. We compiled all node probabilities across our entire data set for each model, and plotted the corresponding cumulative distributions. Interestingly, the ST model has more extremely low-probability nodes than does the LT model, but the overall proportion of these nodes is still extremely small. Neither model assigns any nodes probabilities in the range roughly .001 to .01. Note that from this plot, it appears that the LT model is outperforming the ST model, but this is due to normalization of the proportion of nodes, since there are a large number of nodes with probability near one which are removed from the plot for computational reasons, and in the ST model the number of such nodes is much larger due to floating point rounding in the sigmoid computation.

VI. Conclusion

In this project, we reviewed the Linear Threshold (LT) model, compiled a large ($\sim$ 70 GB) Twitter data set into a real spreading event data set, and analyzed the performance of the widely used and studied LT model. In particular, we found

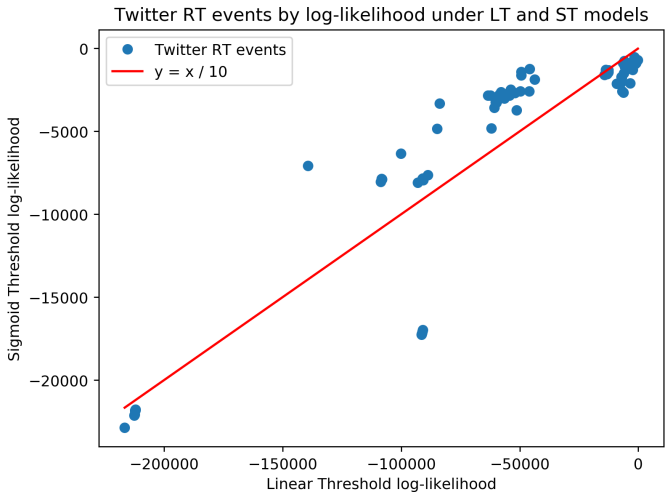

Fig. 8: Log-likelihoods assigned to Twitter RT events by the LT and ST models. Note that the ST values are typically much higher (approximately 10x less negative).

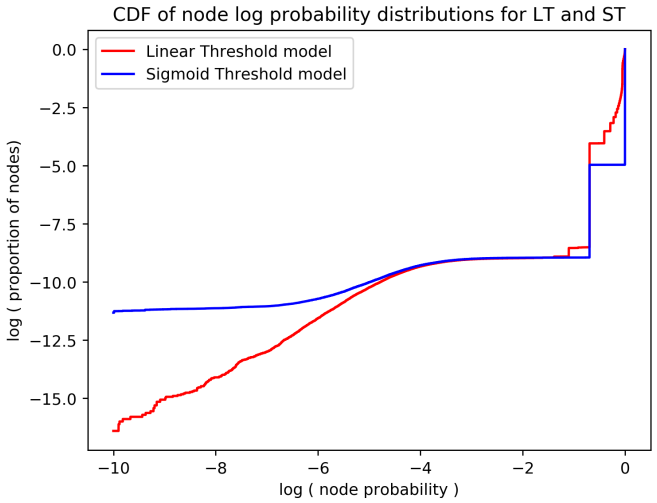

Fig. 9: The CDF of node probabilities under the LT and ST models, compiled across our entire Twitter data set.

that the LT model is not well-suited to modeling our data set, in part due to its problematic restrictions with respect to node in-degree, which we discuss at length in Section V-B.

We then introduced a new model of influence spread, which we call the Sigmoid Threshold (ST) model. The ST model does not directly use node in-degree, so that if one user follows more users who are re-tweeting something, they are more likely to also re-tweet the content, regardless of how many users they are following in total. We show empirically that the ST model has better agreement with real influence spreading processes. Furthermore, our algorithm has global parameters which can be fit to new data sets relatively easily, potentially allowing it to adapt to model spreading processes in dramatically different domains with only a change to these two scalar parameters. This flexibility is lacking in the LT model, except in the choice of edge weights which are constrained by node in-degree.

We also show in Section III-D that the ST model does not directly lead to a submodular influence function, which has significant ramifications with respect to influence maximization algorithms and research. To address this, we propose drawing node thresholds from a specialized distribution, and name a condition on the node threshold distributions which is sufficient to yield a submodular influence function. In other words, we show[4] that for appropriate choice of node threshold distribution the ST model does yield a submodular influence function. We did not have time within the scope of this project to pursue this further, but it is perhaps the most pressing line of future work due to the importance of submodularity in influence maximization.

Once submodularity is established, an interesting direction for future work is to test existing algorithms for influence maximization under our new model. These algorithms have become extremely complex, and we conjecture that they are overfitted to the LT model, such that they will not yield significantly improved performance in practice. A model with better agreement with real data can serve as a valuable testbed for these algorithms, as it may better indicate the generalization performance of methods whose performance has so far only been evaluated on synthetic data generated by the LT model. Of course, there may also be substantial space for new influence maximization algorithms under the ST model, though this is harder to discuss concretely.

References

[1] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 137–146, New York, NY, USA, 2003. ACM.

[2] Pedro Domingos and Matt Richardson. Mining the network value of customers. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 57–66, New York, NY, USA, 2001. ACM.

[3] Wei Chen, Yifei Yuan, and Li Zhang. Scalable influence maximization in social networks under the linear threshold model. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 88–97, Washington, DC, USA, 2010. IEEE Computer Society.

[4] Amit Goyal, Wei Lu, and Laks V. S. Lakshmanan. Simpath: An efficient algorithm for influence maximization under the linear threshold model. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining*, ICDM '11, pages 211–220, Washington, DC, USA, 2011. IEEE Computer Society.

[5] Amit Goyal, Francesco Bonchi, and Laks V.S. Lakshmanan. Learning influence probabilities in social networks. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 241–250, New York, NY, USA, 2010. ACM.

[6] Jaewon Yang and Jure Leskovec. Patterns of temporal variation in online media. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 177–186, New York, NY, USA, 2011. ACM.

[7] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is Twitter, a social network or a news media? In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 591–600, New York, NY, USA, 2010. ACM.

[8] Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.*, 16(5):1190–1208, September 1995.

[4] Or rather, argue; we did not have space to provide a proof, but it is not too complex due to the relative simplicity of the ST model dynamics.

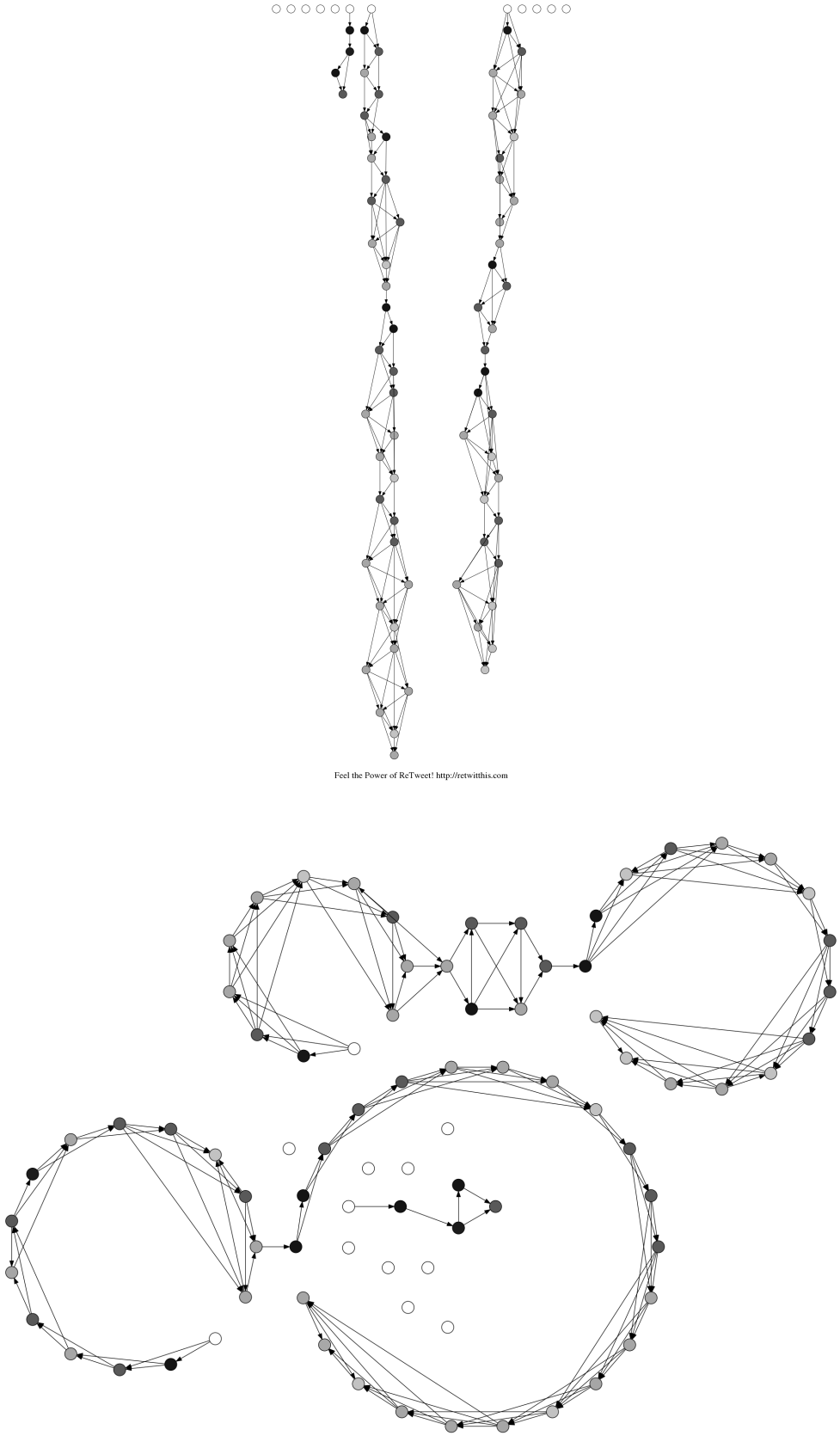

Fig. 10: Spreading process for an unusual re-tweet event, in which users re-tweeted ”Feel the Power of ReTweet!”. The figures are different plotting algorithms’ depictions of the same influence spread network. Comparison with a typical event in Figure 6 indicates the strange pattern of behavior in this event.