Analyzing Twitter Spam

Kanak Biscuitwala, Vidya Ramesh, Kevin Tezlaf – CS 224W Autumn 2011

Introduction

“Tweet," "Retweet," and "Tweeting" were all nonsensical words before 2006 when Twitter was started and became the biggest micro-blogging website. Twitter started off as a social networking site where users would post little tidbits about what they were doing and interesting things that happened to them. Accordingly, the word "twitter" means a short burst of inconsequential information. The idea became so popular that in February of 2010, the site reported about 600 tweets were sent every second [1]. Companies caught on to this popular idea and Twitter expanded to include advertising and as a way to keep in touch with customers. Just as with all popular ideas, Twitter soon became the target of users who wished to misuse their services. Twitter defines this behavior as "spamming" and the tweets that constitute this as "spam". Despite Twitter's best efforts and the efforts of legitimate users to report spammers, spam continues to be a problem in the Twitter network. The advent of URL shorteners created an extra problem for Twitter. Sites like bit.ly, goo.gl, and tr.im allow users to post very short and concise links to websites. As URLs became longer and it became harder to post both a URL and a description of it, users turned to URL shorteners to make it easy to post a tweet describing the URL as well as the URL itself. Twitter must not only check the URL to see if it is from a blacklisted site, but it must also check shortened URLs to see if the eventual redirection results in a blacklisted site.

Due to the popularity of the site, Twitter has been the focus of multiple research projects analyzing information flow and the relationships in the Twitter network. Our paper will attempt to take a unique approach to the Twitter network and study the propagation of spam in the network. Our ultimate goal is to determine if there is a pattern to how spam propagates through the network and if so, then to find a way of determining if accounts have either been compromised and overtaken by spammers or if certain accounts are purely spammer accounts. We will do this by examining the characteristics of the graph of spam tweets as well as running TrustRank on the data that we have. In the process, we may come across other interesting results, such as what characteristics a high-quality user may possess.

Related Work

Twitter Information Flows

In a previous offering of this course, a group presented [6], a report entitled Information Flows on Twitter. The authors explored, among other phenomenon, the outbreak analysis of retweets on a network of Twitter users. They attempted to infer the parameters of a common outbreak model by examining their data and fitting with a least-square distance. However, this analysis treats all tweets equally and does not examine the behavior of spam propagation in isolation.

Twitter URL Blacklists

Grier et al in [2] comment on the effect of URL blacklists on spam detection. They note that blacklists of malicious URLs typically lag behind the appearance of links on Twitter. However, these blacklists eventually “catch up” and add newly detected domains. Because of this, to accurately capture the tweets containing spam URLs, we must use recent URL blacklists, but sufficiently old tweet data. The time required is relatively short: on average, the propagation delay is approximately ten days. Once URLs are tweeted, they appear on future blacklists with high probability. Since our dataset is a snapshot of the past, we can use blacklists effectively to identify untrustworthy URLs and quickly build graphs of spam propagation.

PageRank and TrustRank

Garcia-Molina, et al in [4] introduce an extension to the PageRank algorithm by attempting to incorporate the idea that trust can "flow" across the edges of a network. It essentially introduces a "personalized" version of PageRank where the "teleportation" aspect does not pick a node uniformly at random. Instead, there are a small number of human-curated "seed" pages that can make up an initial vector to create unequal weights for "teleportation" likelihood. The authors, however, focus on web spam rather than Twitter spammers, and they only attempt to identify the most trustworthy nodes, while we would like to identify untrustworthy ones as well. We do intend to follow the same general pattern of evaluation as the authors, however. We plan to use our programmed tools as well as human judgment in order to attempt to identify patterns in a subset of the ranking results.

We also intend to make use of traditional PageRank to help to seed the TrustRank algorithm. PageRank, as described in [9], is a method of ranking nodes in a graph based on how many high-ranking nodes link to a given node. The approach is unique in that it assigns each node a single score that is intended to represent its quality. For both PageRank and TrustRank, an interesting challenge is that we are ranking users that "link" to other users based on the text of their tweets. There is an explicit level of indirection in this case when compared to the web page case. Thus, we will need to adapt our labeling and evaluation methods to compensate.

Data Collection

The data that we are going to use to perform our analysis is data offered by Professor Jure Leskovec and it spans June 2009. The data contains the following information about the each tweet:

- Time (YYYY-MM-DD HH:MM:SS)
- URL (http://twitter.com/author_name)
- Content of tweet (unstructured, but all URLs can be identified by 'http://URL')

The information we are going to find critical is the name of the author and the content of the tweet with particular attention to the links that are provided.

The first tool we used transformed the data into a more usable format. Each tweet with its corresponding information was structured into a row in a Comma Separated Values file. The file was a list of tweets looking like:

author_name, tweet

The name of the author was transformed from its original format to one where the only the author_name was saved.

Next, we created a tool that took the formatted data and striped out any of the tweets that did not contain URLs. After manually examining some of the original data, we determined that all the URLs in the tweets were written in the format "http://URL" or "https//URL". We could not find any URLs in the format "www.URL". We generated a file similar to one previously generated with the tweets containing URLs.

The next tool we created took the tweets that had URLs and created a list of nodes. Every author was considered to be a potential node, but we did not allow authors to be added twice to the list of nodes. Then, we created a tool that took the tweets that had URLs and created a list of edges. A node can have an edge in two ways. First, we defined the concept of retweeting. A retweet is a tweet posted by a user that was posted by another user originally. Retweets are usually in the format "RT @orignal_author the tweet is copied here". If a user retweets another user, an edge is created from the user that is retweeted to the user that retweets. The second way a node can have an edge is if a user mentions another user in his tweet. Users in

Twitter can mention other users in their tweets by adding "@username" to the content of their tweet. The mentioned user then gets notified of the tweet. The edge would go from the user who created the tweet to the user who is mentioned in the tweet.

After the application of all the tools we created on the original data, we were able to create the first graph of our data.

> ***Graph1:*** *A graph G(V, E) in which the set of nodes V represents Twitter accounts involved in at least one tweet (as either the sender or receiver) containing a URL, and in which the set of edges E represents these URL-containing tweets, directed from the author to the tweet's receiver(s).*

Our data comprises all tweets sent and received during the month of June 2009, totaling 18,572,084 tweets in all. Graph1 has 1,048,010 nodes and 2,016,814 edges. Since Graph1 only contains tweets with URLs, this shows that only 10.85% of the original data contained URLS.

We then developed a tool that determined if a tweet was spam or not. As noted in the literature [2], there are several methods to determine the presence of spam, and indeed there are multiple classes of spam including phishing, malware, and scams. For our analysis, we define a tweet as spam if it 1) contains at least one URL and 2) one of these URLs either directly or indirectly (as through redirection) resolves to a domain contained in at least one of a set of spam blacklists. We selected this set from a comparative list of blacklists as found in [3]. In addition, we identified tweets as spam if they contain at least one URL which eventually resolves to a landing page that contains no information (a blank page) or the URL that was redirected to was in a whitelist of URL shorteners. The second identification method was based on the process of URL shorterners to check for spam and then display a page to the viewer of the shortened URL that warned the viewer of possible spam. We initially attempted to run our tool on Amazon's Elastic MapReduce service. However, that approach proved unfruitful as the Hadoop framework that is in place on Amazon's service created multiple problems. We eventually generated the set of spam tweets by making use of our personal computing resources.

> ***Graph2:*** *A graph G(V, E) in which the set of nodes V represents Twitter accounts involved in at least one tweet (either the sender or receiver) marked as spam, and in which the set of edges E represents these spam tweets, directed from the author to the tweet's receiver(s).*

Graph2 has only 124,571 edges and 124,820 nodes. This makes the set of spam tweets to be only 0.6% of the original data and 6.0% of tweets containing URLs. This was expected since Twitter has very efficient spam identification service in place.

Graph Analysis

We successfully generated both graphs after multiple attempts. The links graph contained an average clustering coefficient of approximately 0.0167 while the spam graph contained an average clustering coefficient of approximately 0.0055, roughly 33% of the former. Therefore, as a result of Twitter's spam identification service, Twitter accounts send and receive spam links much less frequently with neighbors of neighbors, a positive sign in reducing spam propagation.

Figures 1 and 2 show the degree distributions of the links graph and spam graph respectively. Both degree distributions exhibit the straight-line signature of a power law distribution when plotted on a log-log plot. A least-squares regression yielded an estimated $y = (-1.0402*10^{-2})*x +1.7049*10^{2}$ for the links graph degree distribution, while a least-squares regression yielded an estimated $y = (-4.0232*10^{-2})*x +1.1571*10^{2}$ for the spam graph degree distribution. Though both plots exhibit power-law behavior, the spam graph clearly decays much faster than the links graph, a result of preventing spam link propagation on the network.

Figures 3 and 4 show the sizes of the ten largest strongly connected components of both the links graph and spam graph respectively; figures 5 and 6 similarly show the sizes of the ten largest weakly connected components of both graphs. Interestingly, while the links graph contains a giant strongly connected component of nodes, the spam graph does not (compare largest strongly connected component size of 130,253 in the links graph vs. 643 in the spam graph, or equivalently 12.43% of the total nodes in the former vs. 0.52% in the latter). Both graphs contain a giant weakly connected component: the largest weakly connected component in the links graph contains 857,512 nodes (81.82% of total), with the largest weakly connected component in the spam graph containing 59,352 nodes (47.55% of total).

These results are plausible considering that normal (i.e. non-spam) Twitter communication regularly involves two consenting parties sending and receiving tweets between each other, resulting in directed edges in both directions. Spam tweets, on the other hand, do not represent such a relationship; in this case, one party (the spammer) produces a directed edge in one direction while the other party (the receiver) usually does not. An analysis taken over hundreds of thousands of these relationships would produce a giant strongly connected component as seen in figure 3 (that is, for mutually consenting tweets), while *not* producing a giant strongly connected component for spammer-receiver relationships (figure 4).

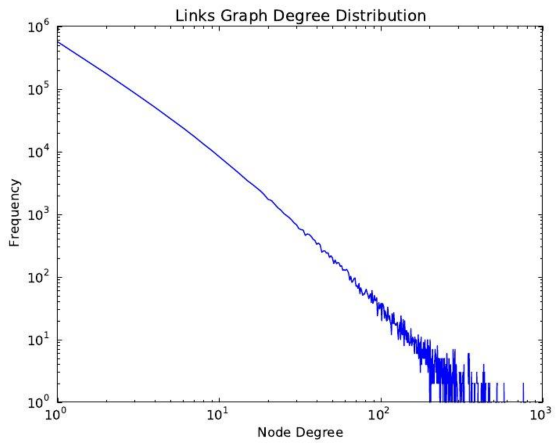

Figure 1. Links graph degree distribution

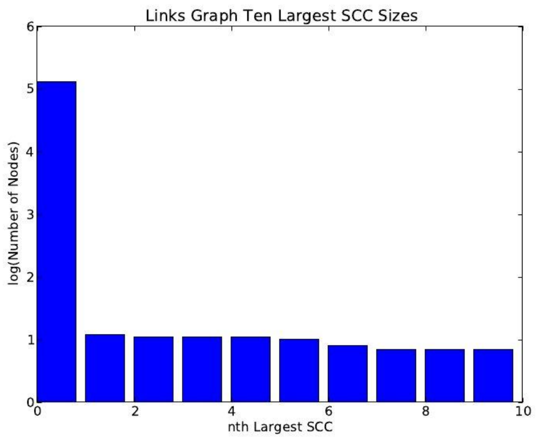

Figure 3. Links graph ten largest strongly connected component sizes

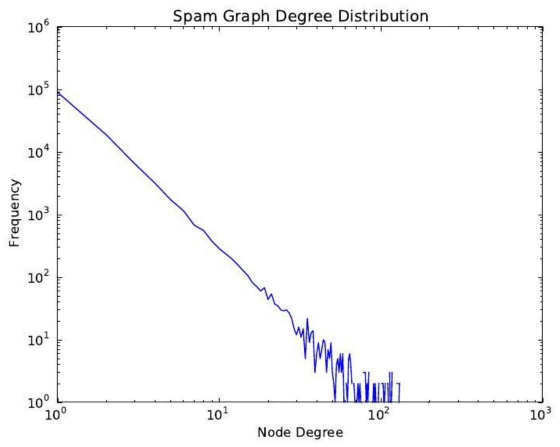

Figure 2. Spam graph degree distribution

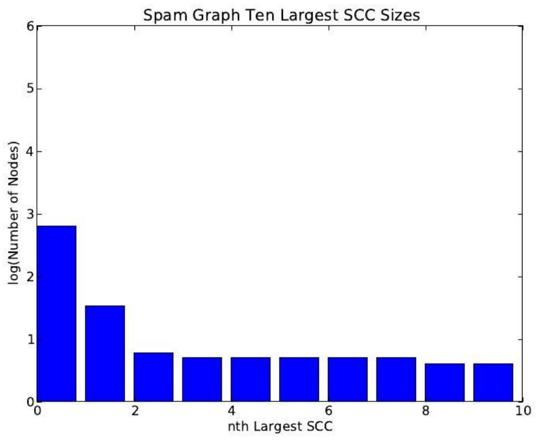

Figure 4. Spam graph ten largest strongly connected component sizes

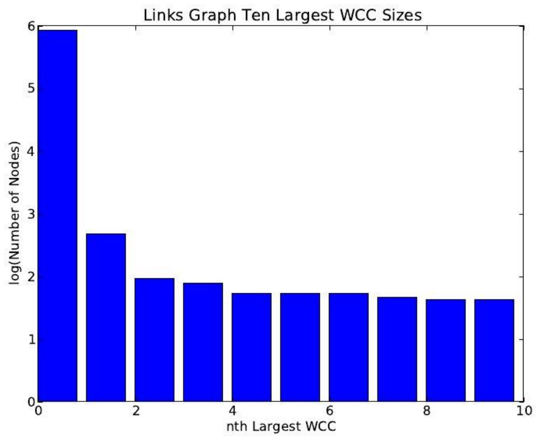

Figure 5. Links graph ten largest weakly connected component sizes

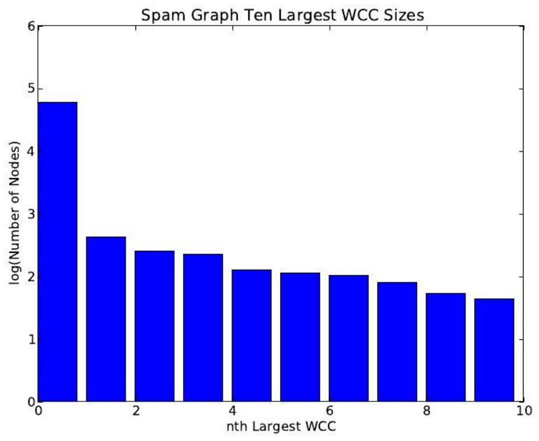

Figure 6. Spam graph ten largest weakly connected component sizes

TrustRank

TrustRank is described in [4] as a variant of Personalized PageRank where a site linking to another indicates that the linking site is putting trust in the linked site. Our model is somewhat different in that the nodes are users who can post URLs that may or may not be trustworthy. The important observation in the paper is that trust should "flow" via the edges in the network. Because of this observation, the graph required as input to the algorithm must be constructed slightly differently from the previous experiments, however the tools that we have already implemented to construct the other graphs can easily be modified to accommodate the construction of this graph.

The TrustRank input graph G(V, E) is formulated as s. V, as before, is the set of all the users present in the tweet dataset. A directed edge from u to v exists in E if u retweets v. Notice that this formulation is the opposite of what was used earlier. This is because we want to measure flow of trust, not influence. Intuitively, if someone retweets a tweet, he is expressing trust towards the original poster. In addition, unlike before, this graph contains tweets regardless of whether or not the they contain links; retweets convey trust regardless of the content of the tweet. Finally, this graph makes no attempt to incorporate regular mentions because a mention could be used for a number of reasons; it may convey trust if the tweeter believes the mentioned user will act on the tweet in some fashion, but spammers may mention a large number of users simply to try to propagate information. Given the potential for noise in result data due to regular mentions, we ignore them entirely as retweets will more reliably convey trust propagation.

Given this graph definition, the network we will evaluate has 548,119 nodes, 986,086 edges, and average in and out degree of 1.7990.

Methodology

The algorithm for TrustRank is described in detail by [4]. Essentially, the authors manually select a small number of trustworthy seed nodes as input to a personalized version of PageRank [5]. The key equation is below:

$$\begin{aligned}&for\ i = 1\ to\ M\ do\\&\qquad \boldsymbol{t} = \alpha \cdot \boldsymbol{T} \cdot \boldsymbol{t} + (1-\alpha)\cdot \boldsymbol{d}\end{aligned}$$

Here, **T** is the transition matrix of the graph, where each column is normalized by out-degree. **t** is initially **d,** a normalized version of a vector of seed trust values; for the first run, an element is 1 if it is one of L seed nodes and is known to be trustworthy. M is the number of iterations of the algorithm; the authors of [4] chose M to be 20 to get sufficiently close to convergence. Finally, alpha is the same decay factor as in PageRank.

Unlike the authors, who focused only on trust propagation, we also want to find nodes which are untrustworthy. Thus, we also want to gauge "distrust" propagation. We can do this by running TrustRank twice: once with trustworthy seed nodes, and then again with untrustworthy seed nodes. Both will yield different trust orderings and values for each node. Because raw TrustRank scores will not be normalized with respect to one another, for each node, we do the following: assign score 1.0 to the node with smallest raw score and assign the ratio of raw score to lowest score for all other nodes. Thus, we now have a meaningful scoring system where the score signifies how much "better" or "worse" a node is relative to another node. Ideally, we want all of the users with negative scores to be ones from which a strong conclusion about negativity can be derived. From Twitter's perspective, the best case result from this ranking is to be able to automatically ban users with a negative score. A slightly weaker usage could be to label of a user's tweets as potentially spam on the user interface. The weakest, but still useful usage is to exclude these users' posts from any "top" tweet listings.

For our purposes, we can add the trust vector from the first run to a constant multiplied by the opposite of the distrust vector from the second run. Then we have an ordering called Hybrid TrustRank that combines influence from trustworthy and untrustworthy nodes:

$$overall\ score\ per\ node = trust\ propagation\ score - \gamma \cdot distrust\ propagation\ score$$

Given the aforementioned trust propagation graph, we must define a reliable method for determining a small set of initial nodes that will serve as the "seeds" from which trust and distrust propagate through the network. This number should be relatively small because each one requires oracle evaluation as to whether or not a node is trustworthy. For a much larger network, the authors of [4] selected fewer than 400 seed nodes. Thus, we can select between 100 and 500 nodes to seed both the positive and negative algorithm.

To select these seeds, we will run PageRank and inverse PageRank to approximate high-quality and low-quality nodes. We have already implemented tools for classifying links as spam, as well as within Alexa's top ranked sites, so we will pick as "good" seed nodes as nodes from the PageRank top 500 which contain reputable links and pass a manual check of a tweet sample. Similarly, "bad" seed nodes will be nodes from the inverse PageRank top 500 with at least 50% of tweets containing spam links and fail a manual check of a tweet sample. Inverse PageRank is equivalent to standard PageRank except that the transition matrix U measures out-links rather than in-links and the "teleportation" vector assumes equal probabilities. Note that the damping factor alpha need not be the same as that used in TrustRank, but for both we will start with 0.85. Note that for the purposes of implementation, this is equivalent to running PageRank on the graph with the edges reversed.

```
$for\ i = 1\ to\ M\ do$
    $\boldsymbol{t} = \alpha \cdot \boldsymbol{U} \cdot \boldsymbol{t} + (1-\alpha) \cdot \frac{1}{N} \mathbf{1}_N$
```

Evaluation

We want to evaluate the following: (1) the effectiveness of PageRank and positive TrustRank in selecting high-quality Twitter users and (2) the effectiveness of hybrid TrustRank in selecting low-quality Twitter users. The utility of the former is that these nodes can be featured prominently on Twitter with minimal need for human curation. The Twitter application for iOS [6], for example, both identifies "top" tweets in a search and users who have recently posted a tweet of interest. Being able to know which users are likely to post high-quality content a priori could improve Twitter's ability to maintain "top" content in real time. The utility of the latter is clear: if a user is shown algorithmically to be untrustworthy, he is a prime candidate for account termination. Even if the resulting level of certainty is not high enough to make such a decision, Twitter can still label potentially untrustworthy tweets as such with user interface elements, and can prevent any such tweets from appearing prominently anywhere, especially not in "top" lists.

Identifying High-Quality Users

For the first stage, the method of evaluation is roughly equivalent to that in [4]. Specifically, we select a sample of size roughly 500, but not uniformly at random because of the sparseness of the graph. Instead, we define 20 buckets where each bucket contains nodes which sum to 5 percent of the total PageRank score; for example, the first bucket contains the first set of elements that sum to the first 5% of the total. These buckets will progressively increase in size since PageRank follows a power law distribution. From each bucket, select 25 nodes at random. Since 500 is a moderately small number of users (especially when compared to the whole graph), we can individually examine these nodes manually to classify them as good or spam for our oracle. For positive TrustRank, we define buckets in the same way, except with the positive TrustRank data.

To mitigate human bias in manual oracle generation, we define a scoring system that assigns a score from 1 to 4 for each Twitter user in the evaluation sample. A score of 1 indicates that the user posts malicious links, mentions irrelevant users, and posts the same outwardly linking tweet repeatedly. A score of 2 indicates a user who uses Twitter as a marketing link aggregator. A score of 3 indicates a user who uses Twitter as a link aggregator with minimal community interaction. A score of 4 indicates a user who engages in relevant personal interaction with the Twitter community. Figure 7 compares PageRank to positive TrustRank by a mean square bucket score s_f defined in terms of the base score s_b and bucket size n_b as follows:

$$s_f = \frac{\sum_{b=0}^{n_b} {s_b}^2}{n_b}$$

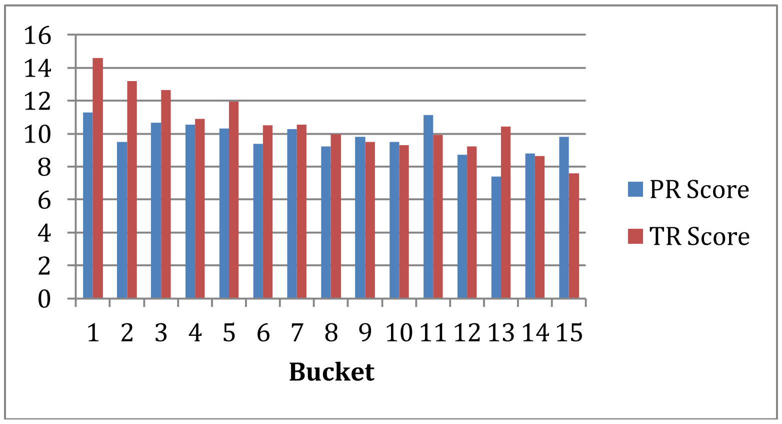

Figure 7. Comparison of PageRank and TrustRank average quality scores per bucket

Evaluation Method	Pearson Correlation Coefficient
PageRank	-0.56956689
TrustRank	-0.899156483
Hybrid TrustRank	0.831440643

Table 1. Correlation between scores for ranking algorithms and bucket number

Table 1 shows the Pearson correlation coefficient for bucket numbers with respect to PageRank, TrustRank, and Hybrid TrustRank as defined previously. A value close to 1 indicates strong positive correlation, a value close to -1 indicates strong negative correlation, and 0 indicates no correlation. Note that Figure 7 omits buckets 16 through 20 because they almost completely contain nodes with zero out degree. These nodes are not influential enough for any reasonable conclusion to be made with respect to their quality. Generally speaking, the graph demonstrates that for both PageRank and positive TrustRank, the overall quality of the higher-ranked buckets is higher in general. Because the scoring is the mean square of the base score, the maximum possible value is 16, and TrustRank on the first bucket approaches this value.

The TrustRank data clearly shows a much clearer decrease in scoring as the bucket number increases; the PageRank data also roughly shows this trend, but it is not as well-defined. More formally, the Pearson correlation coefficient for TrustRank is nearly -0.9, whereas it is approximately -0.5 for PageRank. Thus, based on the evaluation sample, a high positive TrustRank appears to be a strong indicator of Twitter user quality, especially in the case of this evaluation sample.

In order to demonstrate that the strong correlation between quality score and TrustRank data, consider the nodes sampled to evaluate PageRank. Define a "poor" node as one who receives a baseline score of 1 or 2. Then, we can measure the average demotion of a "poor" node in TrustRank relative to PageRank for each PageRank bucket. That is, if a node starts in PageRank bucket b, but is in TrustRank bucket b + k, then the demotion value of that node is k. Figure 8 shows the average demotion of "poor" nodes from each PageRank bucket.

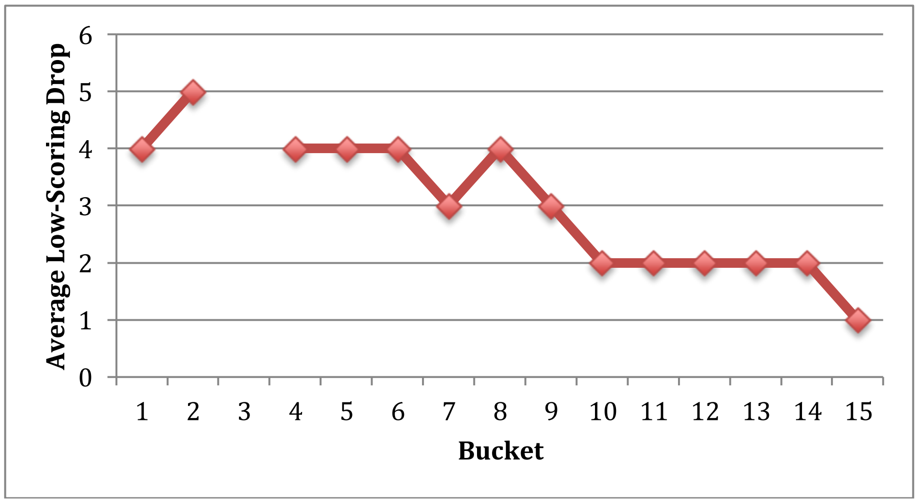

Figure 8. Average bucket demotion for users with base quality score of 1 or 2

Figure 8 demonstrates that "poor" nodes that were previously placed in a high-quality PageRank are more likely to be placed in a much lower TrustRank bucket. Note that there is a discontinuity for bucket 3 because that PageRank bucket did not contain any "poor" nodes. Though every node in the sample experiences some level of demotion, it is promising that the nodes in the buckets representing the highest ranks experience the greatest demotion. This is especially important because, ideally, the top-ranked buckets should only contain nodes of the highest quality, and so any approach that can aid in pruning low-quality nodes from these buckets is valuable. Because of the high level of demotion of low-quality nodes, the remaining nodes are likely to exhibit positive characteristics, and can potentially be used anywhere Twitter would like to highlight "top" users.

Identifying Low-Quality Users

For the second evaluation stage, we examine the Hybrid TrustRank approach as previously defined. First, we chose γ as 0.6 as this produces a large enough number of nodes whose overall score is negative. Also note that we reverse the resulting ranking number so that the largest magnitude negative ranks appear first. For the limited network from 2009 we are evaluating this is roughly 6000 nodes, which is 1 percent of the entire network. Given that this data is only from a single month of twitter usage in 2009, and the Twitter network now gains roughly 400,000 new users each day [8], even isolating one percent of users who exhibit suspicious characteristics. Thus, if we can make strong conclusions about this sample, there is potential for significant impact on the Twitter network. Evaluation buckets are defined as before, except that there are only ten buckets, each representing ten percent of the total score distributed over negative scoring nodes. Positive scoring nodes are ignored entirely in this analysis as these are deemed to be more trustworthy than untrustworthy, and thus should not be subject to any repercussions. Figure 9 shows the average mean square quality score of the evaluation sample as a function of the bucket.

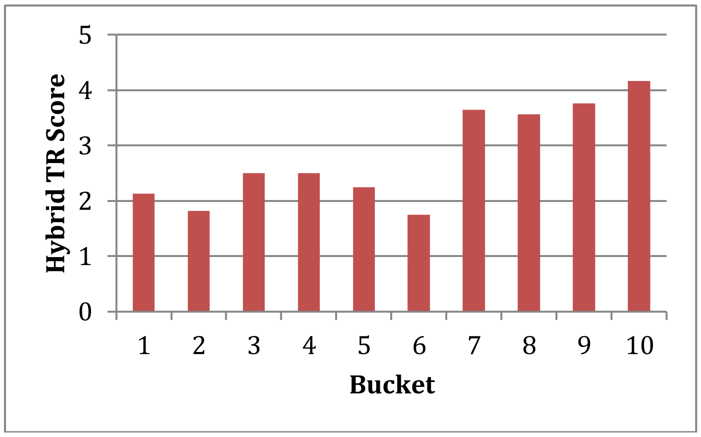

Figure 9. Average quality score for each hybrid TrustRank bucket

Notice that the mean square quality score can range from 1 to 16, and the highest scoring bucket only receives approximately 4. Thus, the great majority of these nodes are "poor" nodes with base score of 1 or 2, indicating that highly-ranked Hybrid TrustRank users are likely to abuse their Twitter user privileges. Visually, it is also clear that the highest ranking buckets contain the lowest-scoring nodes. Table 1 also shows that the Pearson correlation coefficient relating bucket to quality score is 0.83, indicating that the Hybrid TrustRank metric is effective at selecting the most untrustworthy nodes from the retweet network.

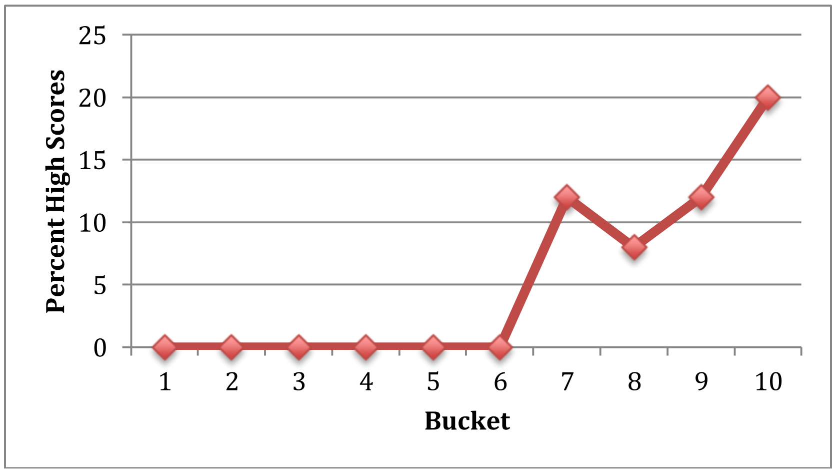

Figure 10. Percent of users with base score 3 or 4 in sampled hybrid TrustRank buckets

An important consideration to make when attempting to label nodes as undesirable is the likelihood of false positives. Figure 10 plots the proportion of false positives in each Hybrid TrustRank bucket. For the first six buckets, there were no false positives in the evaluation sample. For the remaining buckets, the false positive rate increases with the bucket number, but is at most 20 percent. Essentially, what this data suggests is that nodes that behave suspiciously enough to receive the poorest Hybrid TrustRank scores deserve termination with high probability. Because TrustRank follows a power law distribution, there are many more nodes in the final four buckets as compared to the first six buckets. These can still be labeled by Twitter as potential sources of spam and the tweets in search results and elsewhere can be labeled as untrustworthy.

Overall, this approach is a high-speed method of identifying the most untrustworthy nodes with reasonably high certainty in the large Twitter retweet network. Combined with existing approaches, such as machine learning on actual tweet text, this can serve as a powerful tool for neutralizing spam on Twitter.

Communities within the Graphs

We used the Louvain method in order to detect communities in our graphs in order to determine if we can combine what we know trustworthy nodes and communities to identify spam communities. The Louvain method is a greedy optimization method that attempts to optimize the modularity of a partition in a graph [7]. Modularity is defined as how well divided a network is into particular partitions. Networks that have less edges between partitions and a high density of edges within partitions have higher modularity than networks that have a high density of edges crossing partition boundaries.

```
def louvain_method(graph) {
    while(modularity != optimal) {
        small_communities = optimize modularity locally
        for each community in small_communities:
        all_communities += community represented as node
        graph = all_communities
     }
    return graph
}
```

The method starts off by optimizing modularity locally resulting in the formation of small communities. The small communities are aggregated together and represented as nodes in a new graph in the next step. The method repeats these two steps until the modularity is maximized and the graph is created with communities and sub-communities. Exact modularity optimization is a NP-hard problem and this method, in contrast, is efficient and easy to implement. It cannot guarantee perfect community detection, but it is a very good approximation.

The use of this method on our spam graph proved very interesting. We noted that there were 26081 communities within the spam graph. There are a high number of communities of size 2 or less which would suggest that all the spammers in the Twitter network are individuals working outside of communities. Even though there are many more communities of two nodes than communities of larger sets of nodes, only 16% of nodes are in these smaller communities. The remaining 84% of nodes are part of communities that are larger.

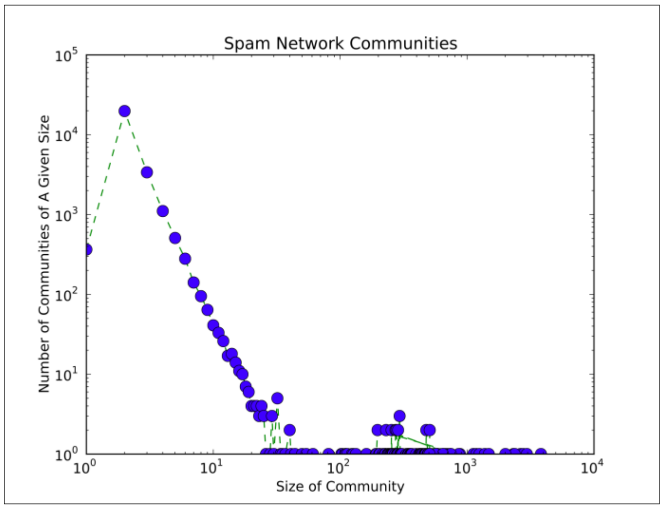

Figure 11. The number of communities of a given size in the spam network

The largest community we detected in the spam network was 3820 nodes and only 12 communities have more than 1000 nodes in them. The trend of communities in the spam network comes as no surprise when the trend of the communities in the network of tweets containing URLs is examined. Both the spam network and the URL network have the same trend of a large number of small communities and fewer large communities.

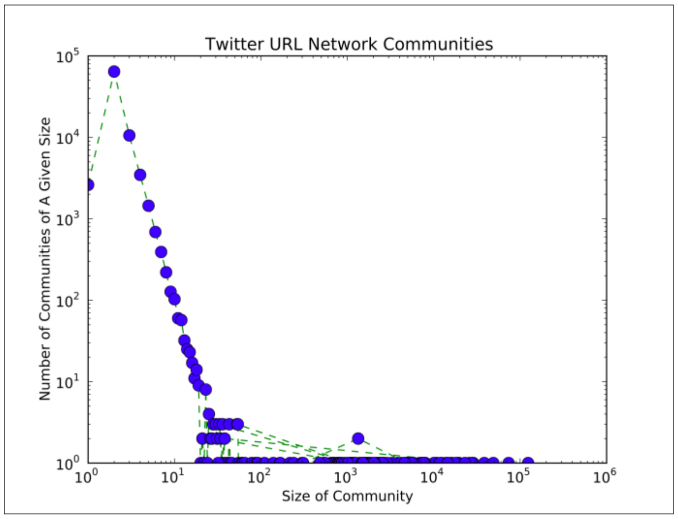

Figure 12. The number of communities of a given size in the network of tweets containing URLS

We also noted that while spammers act within communities, most spammers do not exclusively create spam tweets. We sorted users into three different buckets. The first bucket we called "pure spammers" and these users were spammers who only created spam content. The second bucket was called "more than half spammers" and these were spammers who created spam content more than half of the time. However, this bucket of users did not include users who were "pure spammers". The final bucket we called "less than half" and these spammers only tweeted spam tweets one in every two tweets or less. These definitions helped us analyze our spam graph to note which bucket the majority of our spam users identified with.

Only 42852 users or 34.33% were "pure spammers". In contrast, 77341 users or 61.96% of spammers fell into the bucket of "less than half". Interestingly, the bucket of "more than half" barely contained 3.71% of users. There are a couple of possible reasons for this unusual division among the buckets. We hypothesize that spammers on the Twitter believe that if they tweet spam tweets more than half of the time, it is just as likely to get identified as a spammer if they are tweeting spam tweets all of the time. This is because Twitter is very good at detecting spammers who only create spam tweets and users who tweet spam more often than not. Our dataset might also be influenced by the fact that Twitter has methodologies in place for detecting and removing spam users. This could influence the number of users who tweet spam more than half of the time. However, we would expect to see that this affects the percentage of users who are "pure spammers". Since we don't see this, we are unsure on how the Twitter's spam policy has influenced our dataset. The fact there are significantly more spammers who tweet legitimate URLs along with spam URLs does suggest that the latter reason could be a great motivation for spammers to appear legitimate at least half of the time if they wish to continue to create spam tweets.

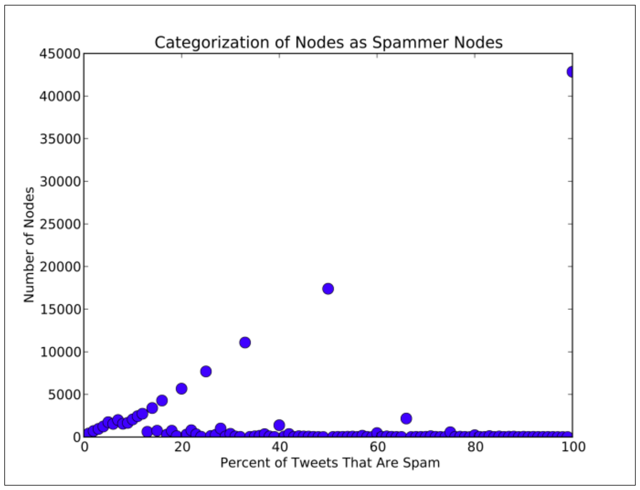

Figure 13. Number of users who have a given percent of tweets defined as spam

We attempted to identify which communities these “pure spammers” belonged to in order to detect any possible pattern. If the “pure spammers” congregated together in the larger communities, this simplified Twitter’s identification of spam problem. Twitter would then have to only identify key spam users and the communities they belong to. Twitter could take the appropriate action to fully determine if those communities are spam communities and warn legitimate users of spam from those communities. 16495 communities contained “pure spammers”. This is about 63.24% of all the spam communities. Figure 14 shows that the pure spammers aren’t confined to communities of particular sizes. In fact, they seem to be evenly distributed with no bias towards larger or smaller communities. Unfortunately, this does imply that Twitter cannot take advantage of the size of the community when attempting to eradicate pure spammers.

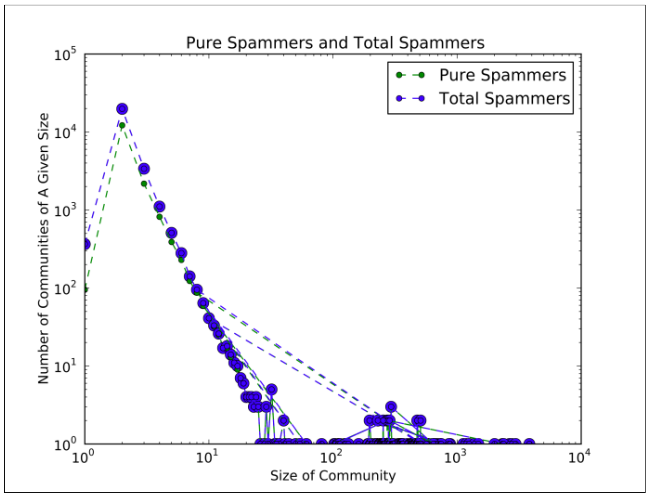

Figure 14. The sizes of communities containing “pure spammers” compared with the sizes of all communities in the spam network

Further Work

The TrustRank analysis yielded some surprisingly favorable results and effectively separated high-quality users from low-quality users. However, false positives certainly existed and before Twitter is able to incorporate methods such as ours, we must reduce the false positive probability as much as possible. We propose a multi-phase approach of first identifying high- and low-quality users using positive and hybrid TrustRank approaches, followed by some machine learning and natural language processing methods to analyze the actual text of the users’ tweets. One possible approach is to train a classifier on the massive existing database of spam email and apply the classifier to individual tweets. A user identified by hybrid TrustRank that scores poorly on this classification phase is a spammer with high probability.

Similar approaches could also be applied to our analysis of connected components. We can identify strongly and weakly connected components, run a form of analysis on large groups of nodes that form the latter, but not the former. Thus, we reduce the evaluation size for even the most computationally-intensive classifier. Any of the methods we have presented have the potential to aid in improving the performance of more fine-grained methods by identifying key parts of the network on which to focus.

Conclusions

In our attempt to analyze the spam propagation in the Twitter network, we were able to come up with a reasonably efficient method of identifying the most untrustworthy nodes with reasonably high certainty in the large Twitter retweet network. We also showed how while spammers tend to work in communities, those communities lack an easily definable characteristics that we can take advantage of. We also noted that the spam network within the larger Twitter network lacked some of the characteristics of the larger network. The spam network did not contain a giant strongly connected component while the Twitter network did. We considered this a reasonable deduction considering that non-spam Twitter communication involves two consenting parties that would show up as closely linked on our graph. Spam communication tended to follow a spammer-receiver relationship.

The method of scoring high-quality nodes based on positive TrustRank and low-quality nodes based on hybrid TrustRank yielded surprisingly favorable results. Both methods are able to categorize Twitter users in such a way that their ordering is highly correlated with the quality of their tweets. We were able to accomplish this by manually labeling only a few hundred seed nodes, and adjusted the weights of the algorithm as necessary. We were able to consequently identify a fraction of the Twitter user base that could be regarded as untrustworthy. Combined with textual analysis of tweets, Twitter can potentially use this approach to quickly eliminate many of its malicious users.

With respect to our ultimate goal, we have come to the conclusion that while we are able identify untrustworthy users, it would take extensive further work to determine if accounts have been either compromised or created as pure spammer accounts on a purely automated basis. We believe that we have set up the groundwork for others to pursue this as further line of research.

References

[1] http://www.telegraph.co.uk/technology/twitter/7297541/Twitter-users-send-50-million-tweets-per-day.html
[2] http://www.icir.org/vern/papers/ccs2010-twitter-spam.pdf
[3] http://www.sdsc.edu/~jeff/spam/Blacklists_Compared.html
[4] http://www.cs.toronto.edu/vldb04/protected/eProceedings/contents/pdf/RS15P3.PDF
[5] http://www2002.org/CDROM/refereed/127/
[6] http://itunes.apple.com/us/app/twitter/id333903271?mt=8
[7] http://iopscience.iop.org/1742-5468/2008/10/P10008/
[8] http://blog.twitter.com/2011/03/numbers.html
[9] http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf