CS 224W: Spamming PageRank-like Networks

Dan Gnanapragasam (nanabyte@)

Abstract

Link spam is been a serious problem in PageRank-style systems. In this method of spamming, spammers create a collection of *boosting pages* in order to improve the PageRank of one or more *target pages*, artificially increasing its ranking and displacing high quality content pages in the overall ranking of web pages.

In this paper, we look further into the problem of link spam and how we can remove it from the web. In particular, we analyze the process of generating spam farms and then characterize how we could detect them in real life. We propose a variety of spam farms from a purely theoretical basis, propose simple measure to counteract them and then show how spammers can easily thwart these efforts.

By examining network properties of a 2009 snapshot of the Wikipedia graph, we propose a data-driven method to determine if a given page is a valid content page or a spam page. At the core of this method is **out-degree distributions of in-neighbors of a given page**. We show that this distribution is *log-normal* with parameters in a certain range for nearly all of the top 100 Wikipedia pages by PageRank, while none of the standard spam farms fit a log-normal distribution. Lastly, we show that, in order for a spammer to force a misclassification of one of their target pages, they must create a huge number of boosting pages, resulting in between a $1.7x$ and $3x$ increase in the number of boosting pages a spammer needs to create versus the number of pages that they would need to create in one of the ideal spam farms. When coupled with the existing Spam Mass Estimation algorithm, this forces spammers to hijack a significant amount of PageRank from trusted sources and then also generate several useless boosting pages, thereby doubling or tripling the operating costs.

Introduction

PageRank[2] and techniques based on PageRank were popularized in the late 90s and are still used today to rank web content. Many of the existing papers focus on ranking improvements based on biasing the search paths towards topics[6] for relevance or personalization or toward trusted parts of the web[5]. More recent papers have focused on detecting high PageRank spam on the web[3] so that it can be deweighted and on how efficient spam networks are constructed[4].

Spam, however, remains a problem on the web. The "Prior Work" section will show that most method of spam detection rely on "known good" parts of the web and then use network wide value (such as variants of PageRank) to rank the goodness or spaminess of pages. The motivation for this paper is to **determine which, if any, local properties on a network can be used to differentiate spam pages from valid content pages**. Our measures of the effectiveness of the algorithm will be (1) classification accuracy (including false positives) and (2) the increase in cost for a spammer to achieve a spam network with the same PageRank for its target nodes.

This paper is organized as follows: First, we will summarize prior work on this subject and define our motivation. Then, we will discuss simple, efficient spam farms, naive methods to detect these farms, and how spammers can quickly circumvent these methods. Next, we will present network properties calculated from a 2009 snapshot of the Wikipedia pages[1], and provide an efficient algorithm for computing these network properties on a giant graph (5.7 million nodes and 130 million edges), and show how these properties provide a clear classifier between spam and non-spam pages. Lastly, we show that this method of detection forces spammers to generate between $1.7x$ and $3x$ extra boosting pages in order to trick the classifier and still achieve their desired PageRank.

Prior Work

There has been plenty of work on the topic of PageRank, beginning in 1999 with the original PageRank paper[2]. In 2002, Topic Sensitive PageRank[6] was proposed to avoid overweighting high PageRank pages. In 2004, TrustRank[5] was developed to handle the case of link spam. In 2005, optimal spam farms and alliances between spammers were analyzed[4]. In 2006, TrustRank and PageRank were used in tandem for Spam Mass Estimation[3], which provided an efficient method for finding and deweighting high PageRank spam page.

PageRank

In [2], Brin and Page propose and algorithm for ranking content in an unsupervised setting. Previously, web surfing began with "high quality human maintained indices"[2], which were hard to initially construct, difficult to keep current, and poorly scaling for non-head queries. The algorithm of Page and Brin relies on quickly and repeatedly crawling the web to keep fresh versions of the web. Thus any ranking model they use will operate on fresh data, which handles the issues of keeping the index current.

The relevant part of the paper comes from the actual ranking model, PageRank. PageRank models the web under the *random surfer model*, which treats the web as a graph where the nodes are web pages and the directed edges are hyperlinks from one page to another. The surfer begins at a random page and continues clicking links at the given page. With some (small) probability, the surfer will become bored and request a page at random. This can be modeled as a Markov chain where at time t:

$$P(\text{in state } p_j) = \frac{1-\beta}{N} + \beta \sum_{p_i \in n(p_j)} \frac{P(\text{in state } p_j)}{|\text{out-links of } p_j|}$$

where $1-\beta$ is the "teleport probability", N is the total number of pages on the web and $n(p_j)$ is the set of in-neighbors of p_j. Though not explicitly stated in the paper, PageRank computation can be easily parallelized (across a given iteration) and the iterative process will converge in a reasonable amount of time.

The obvious weaknesses come from the fact that this algorithm is not well-equipped to handle spammers. At the time, this was a reasonable assumption to make because the cost of creating several spam pages was too high to make it reasonable to create a spam farm that would challenge the high PageRank queries. However, this is no longer really a problem due to the cheap cost of making a web page. Purely from a ranking standpoint, it also suffers from the fact that there is only a single PageRank value for each page, independent of the query. This isn't an accurate modeling assumption, since a page like cs.stanford.edu might be a great authority on computer science, but might have no real understanding of an unrelated topic, such as art history or football. Fortunately, this problem was solved in [6].

Topic Sensitive PageRank

In [6], Haveliwala proposes Topic Sensitive PageRank to solve a limitation / bad modeling assumption of the PageRank algorithm. As stated above, PageRank creates a single PageRank value for each page. However, this is not optimal, since PageRank is a proxy for the authority of the page and this intuitively should vary depending on the topic. The solution proposed by Haveliwala is to generate a set of PageRank vectors depending on the topic. Then at query time, the appropriate PageRank vector can be used to help rank pages (if applicable).

From an implementation standpoint, Topic Sensitive PageRank is similar to the original PageRank algorithm. At the beginning of the algorithm, 16 topic sensitive PageRank vectors are generated where each vector is the PageRank for a hand-picked set of "basis topics" taken from the "Open Directory". Then, when the query is being calculated, the query is compared against all 16 vectors and a linear combination of the Topic Sensitive PageRank vectors weighted by the similarity is added to create the overall rank for the page for this query.

The computation of these basis vectors is analogous to the vanilla PageRank with the change that the teleport set of pages is restricted to a predetermined set of pages.

The major drawback of this model is that it requires human intervention to create the 16 sets of basis pages. This is bad from a temporal staleness standpoint since authorities on subjects change over time, but it also doesn't scale well to a wider range of subjects (you cannot increase the size of the basis), and it will not allow teleportation into other "good" pages that had the misfortune of not being present in the initial teleport set. Also, it is not immediately obvious that this will punish spammers. Granted, we will no longer teleport into spammer owned nodes, but there doesn't seem to be any active punishing or deweighting of spammy nodes. Fortunately, this was addressed with TrustRank[5].

Trust Rank

In [5], Gyöngyi et. al. actively attack spammers by their formulation of a PageRank like algorithm. In their algorithm, the pick a set of pages that are known to be reputable and good. From here, they run Topic Specific PageRank with the teleport set equal to this trusted list of pages. The meat of the paper comes down to finding the trusted set of pages for the teleport set. One method to find this is inverse PageRank, which comes from inverting all edges in the graph and the computing the PageRank (i.e., you are looking at pages that point at a lot of other pages rather than pages that are pointed at by a lot of pages). However, Gyöngyi et. al. also propose high PageRank as a feature for finding a trusted page, as high PageRank pages will tend to point to other pages with high PageRank, meaning that the property of trustworthy pages pointing to other trustworthy pages comes for free.

The appealing parts of this algorithm come from the use of existing algorithms. PageRank features three times (for PageRank, inverse PageRank, and the actual TrustRank computation). Thus, many of the results and the intuition behind PageRank and Topic Specific PageRank remain usable.

The major drawback of this method was that it still doesn't show how to deal with pages that have a high PageRank, but are spammy. Granted, using verified "non-spam" pages as a seeds will help weed out bad pages as seen by Figure 10 in [5], but these pages will still might end up in the low numbered TrustRank buckets from Figure 10. Moreover, the selection of "non-spam" seeds treats these pages as infallible, meaning that the owners of these pages can use their new power to boost the PageRank of their pages without any checking mechanism (i.e., by pointing at pages that they have created or by cutting links to their rivals).

Link Spam Detection

In [3], Gyöngyi et. al. continue where they left off in the previous paper and try to measure the impact of link spam upon a page's ranking. The motivation comes from the fact that the cost barrier to link spamming dropped substantially in the early to mid 2000s. Gyöngyi et. al. want to quantify the amount of PageRank for a page accumulated from spam pages. The algorithm boils down to computing the PageRank and the core-based PageRank (which is like TrustRank except that instead of using a very small, high quality seed set, this algorithm makes the teleportation set as large as possible while still containing good nodes) and noting that the absolute spam mass is just the difference of these vectors. The algorithm relies on partitioning the graph into a set of good nodes V^+ and a set of spam nodes V^-. The paper discusses choosing a good seed set (called the "good core") and settles on randomly sampling the network and hand-labeling spam vs not spam to get a fraction of the network that is spam and using this to scale the vector for random jumps.

Once spam mass has been computed, the algorithm proceeds in the obvious manner to compute the relative spam mass for all pages (which is the absolute spam mass divided by the PageRank), filters out all pages below a given PageRank threshold (to only penalize high PageRank spam candidates) and finally uses a decision threshold on the relative spam mass to classify spam versus not spam.

This algorithm is very appealing for several reasons. As with TrustRank, it reuses PageRank and existing ranking algorithms extensively. However, the thresholding idea is highly effective, as it ensures that we only make spam / not spam classifications on the pages that users are likely to see. Additionally, it avoids cases where very small PageRank and misestimates of core-based PageRank scores can cause a good page to look like spam.

The main drawback of this algorithm is similar to the drawback of TrustRank. The method of creating the good set of nodes is still a bit hand-wavy and requires manual intervention. Moreover, while the algorithm correctly knocks down spammy nodes, it doesn't generalize well to thwarting a spammer who downsizes their operation to just below the PageRank threshold. Such a spammer will probably use a scaled-down version of their original spam farm, which we should be able to detect via some algorithm. Lastly, the algorithm requires a full run of the TrustRank algorithm across the entire web. Ideally, a detection algorithm should be able to get a good idea about the spaminess of a page using local information about that page. As such, we analyze common spam graphs in the next section to gain insight about their structures.

Basic Graph Schemes

Many efficient spam graph structures come from Gyöngyi et. al.[4]. Define the *target page* to be the page the spammer wishes to boost in PageRank. The spammer also controls several *boosting pages* which are used to increase the PageRank of the target page. Lastly, there are *hijacked links*, which point from the independent web pages to pages controlled by the spammer.

In this paper, optimality conditions are defined for a spam graph. Theorem 2 from Gyöngyi et. al.[4] states that the PageRank score of the target page is maximal iff:

1. All boosting pages point at the target page.
2. No boosting page points to any other boosting page.
3. The target page points to some or all boosting pages.
4. All hijacked links point to the target.

Note that, as a corollary to Theorem 2, in an optimal spam farm, no page in the spam farm can have any outlinks to the rest of the web or else some of the PageRank will escape from the spam farm back into the rest of the web.

Given this fact, we will analyze several spam farms, including those proposed by Gyöngyi et. al. After proposing each type of spam farm, we will present an algorithm that can detect such a spam farm and the show how a spammer can easily work around these naive methods.

Single Target Spam Farm with Reciprocated Edges

The most obvious spam farm consists of a single target page with reciprocated edges between the target page and all boosting pages. If there are k boosting pages and the target page gets λ PageRank from hijacked links, then the PageRank for the target page is

$$\frac{k\beta + 1}{(1+\beta)N} + \frac{\beta\lambda}{1-\beta^2}$$

This is quite efficient. For $\beta = 0.85$ (a commonly used value), 46% of the total PageRank in spammer owned pages ends up in the target page.

However, detection of such a spam farm is incredibly simple. This is because the graph contains a lot of regularity. In particular, all edges in the graph are reciprocated edges. Thus, an easy detection algorithm would involve looping over all nodes with out-degree 1, finding all reciprocated edges that end at these nodes, and tagging all nodes with more than L such reciprocated edges as possible spam target pages. This method of detection follows the intuition that pages with out-degree 1 are only present to boost the importance of their single outward pointing neighbor.

Multiple-Target Spam Farm with Reciprocated Edges

There are several ways a spammer could workaround our detection mechanism. The easiest option would be to create multiple target pages and then add reciprocated edges between the boosting pages and the target pages. The PageRank of the two target pages will be equal. If there were K total boosting pages and two target pages, then the total PageRank of each target page is

$$\frac{\frac{K}{2}\beta + 1}{(1+\beta)N} + \frac{\beta\lambda_i}{1-\beta^2}$$

where λ_i is the PageRank going in to the i^{th} target page. Note that this means that the spammer must double the number of their target and boosting pages, but then achieves two pages with PageRank each equal to the PageRank in the previous question

Once again, detection of such spam farms is made simple by the regularity of the graph. As all edges are reciprocated, we can look for all nodes of degree no greater than 2 and then find nodes which have a high number of reciprocated edges with such nodes. In fact, this argument applies for any T: If the spammer has T target pages, we can search for all nodes of degree at most T and find other nodes where the number of reciprocated edges is above a fixed threshold. As long as T is somewhat small, such pages are a reasonable indicator of spaminess, as it is very unlikely that a reputable page will have several reciprocated edges. However, as T grows large, this method will have more false positives, which makes it unusable.

Eliminating Reciprocated Edges With Intermediate Nodes

Given that our detection methods have relied on finding the reciprocated edges, it stands to reason that spammers will create a graph without reciprocated edges. One possibility is to add intermediate boosting nodes. The resulting structure looks like a tree rooted at the target page. The target pages points to k intermediate pages, which each point to m other pages, which in turn point to the target page. The resulting PageRank of the target page is approximately equal to:

$$\frac{\lambda}{1-\beta^3} + \frac{\beta}{1+\beta+\beta^2}\frac{m+k\beta}{N}$$

where N is the total number of pages on the web. In particular, for $\beta = 0.85$, this means that the PageRank of the target will be close to $\frac{3}{10}$ of the fraction of the total PageRank owned by the spammer (in addition to any PageRank coming from hijacked links).

While this structure lacks reciprocated edges, it intuitively appears to be less effective than the previous structures. Gyöngyi et. al.[4] note in their paper that if the amount of PageRank coming from the rest of the web is fixed and the spam farm has no out edges into the rest of the web, then the PageRank of that spam farm is also fixed. Intuitively, this graph will be less effective than other spam farms because there are several boosting pages that will collect PageRank that could otherwise be diverted to the target page.

On the other hand, detecting such a graph would be slightly tricky. Instead of looking for reciprocated edges, we are now looking for directed triangles. In our spam graph, the target node is present in the highest number of directed triangles. This is analogous to the reciprocated edges computation from the previous graph in that we are really looking for small length cycles in our graph. However, as the number of nodes in the cycle increases, the number of false positives should also increase, as there certainly will be cycles in a non-spam web graph.

Ring Shaped Graphs

To increase the potency of the previous network, the spammers can generalize the concept of intermediate nodes and use the ring-like spam network devised by Gyöngyi et. al in [4]. In this network, there are k target pages that form a singly or doubly linked ring. Each page has some number of boosting pages, b_i which have a single directed edge to the i^{th} target page. Here, the PageRank is boosted within the ring of target pages, while the boosting pages store almost no PageRank.

Detection for these networks is a bit different from previous networks. In particular, the "leaf" nodes in this spam network will have in-degree 0 and out-degree 1. In general, pages like this probably should not influence the PageRank computation, so we can ignore these pages as a post-filtering phase after the crawling phase. Once this is done, PageRank can be computed in the standard manner. However, spammer can easily work around this by adding a small number of pages that each point to the boosting pages. These pages ensure that the boosting pages have a non-zero in degree (so they will not be filtered), but also do not decrease the overall PageRank of the target pages as their insertion does not take any PageRank away from the remaining nodes.

From these simple workarounds to naive detection methods, it is clear that we must look at network properties that are present in an actual, non-spam web graph and determine how these differ from spam graphs.

Real-Life Data, Experiments, and Results

In order to analyze real life data, a 2009 snapshot of Wikipedia data[1] was used for network analysis. This network has ≈ 5.7 million nodes and ≈ 130 million edges. This is not an entirely fair or representative portion of the web, as it has a much high link density that other portions of the web as Wikipedia pages often link to each other. On the other hand, the quality of Wikipedia pages will be pretty high due to the moderating community, who work to remove page vandalism. As a result, our top ranking pages should, for the most part, be valid content pages and not spam.

The essential idea is to find network properties found in real web graphs that would be difficult for spammers to duplicate without either creating several spam pages or hijacking lots of high PageRank links. This immediately ruled out the *diameter* and the *size* of the *weakly connected component*, which are incredibly easy for the spammer to spoof. With only one hijacked link, a spammer could connect a real world network to the spammer's network. As such, the diameter and the size of the weakly connected components for the two networks would now be the same. From here, the most logical next properties to consider were the in and out degrees of the nodes.

In and Out Degree

As an initial step, the in and out degree distributions were computed for all 5.7 million nodes in the Wikipedia graph. Due to the size of the graph, and the obvious parallelism, a MapReduce style algorithm was run in order to generate the desired outputs. The input to the MapReduce was an out-edge adjacency list representation of a graph and the three outputs were (1) an inverted adjacency list, (2) a key-value map of nodes to out degrees, (3) a key-value map of nodes to in degrees. The implementation used a proprietary version of FlumeJava (which is similar to Apache Crunch).

Figures 1 and 2 below show the In-Degree Distribution and the Out-Degree Distribution for all Wikipedia nodes.

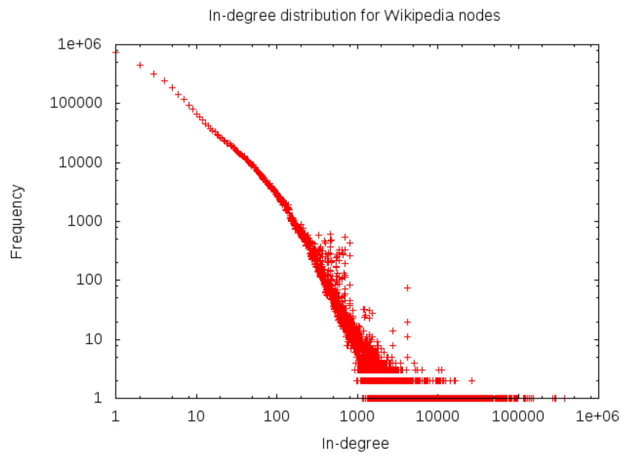

Figure 1: In-Degree distribution for all Wikipedia nodes

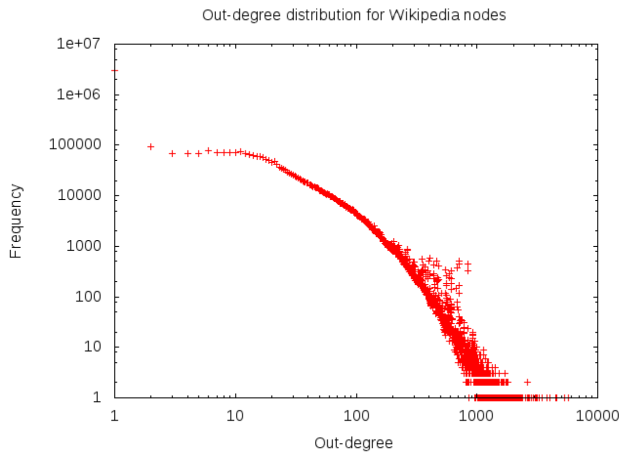

Figure 2: Out-Degree distribution for all Wikipedia nodes

In particular, both distributions fit a power-law, which meant that we cannot easily rule out any node simply by its degree as there will be nodes with very high and very low degrees in either power law.

Top 100 Pages

To make problem more tractable, we only consider the top 100 pages by PageRank, which were specified along with the data source[1]. The in and out degree distributions for these pages are shown in Figures 3 and 4. While the in-degrees still fit a power law, the out-degrees don't fit a clear distribution between 1 and 2000. This gives us a first pass at a simple method to detect possibly spammy pages among high ranking spam pages: Pages with high PageRank, but low in-degree should be marked for further investigation.

Of course, by itself the in-degree is a bit of an easy metric to work around simply by adding more boosting pages. Its not unreasonable to think that a spammer might create 30,000 spam pages (equal to the in-degree of the 100^{th} node by PageRank). A naive way to fix this would be to look at the in-degree graph and state that the ranking of in-degree and the relative ranking in PageRank plots is rarely more than 5 spots out of order. However, a spammer using the single-target, reciprocated edges style graph could easily work around this by noting that a boosting page with $m-1$ additional out edges (to non-target pages) contributes $\frac{\frac{1}{m}\beta+1}{(1+\beta)N}$ to the PageRank of

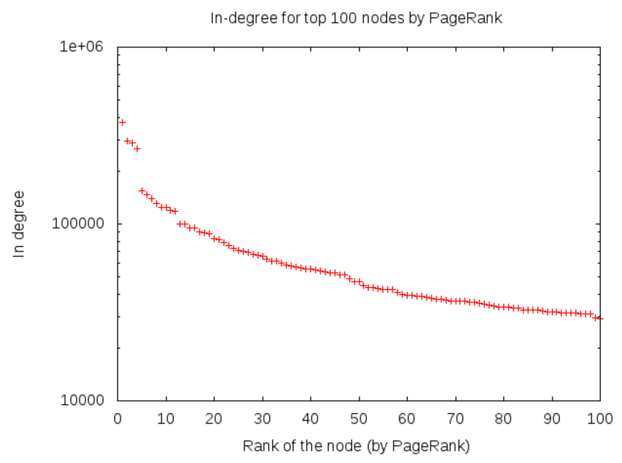

Figure 3: In-Degree distribution for Top 100 Wikipedia nodes by PageRank

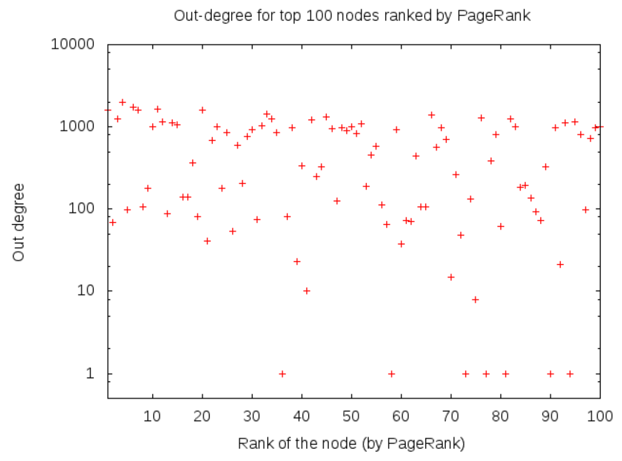

Figure 4: Out-Degree distribution for Top 100 Wikipedia nodes by PageRank

the target page. This leads to the following unfortunate lemma, whose proof comes trivially from this fact.

Lemma 0.1. *If k desired number of boosting pages and λ is the PageRank flowing into the target page, then for any Z where Z can be written as the sum of k positive unit fractions, a target page of can be generated with PageRank $P = \frac{Z\beta+1}{(1+\beta)N} + \frac{\beta\lambda}{1-\beta^2}$.*

Neighboring Degree Distributions

The failure of the previous detection mechanism suggests that we need properties that are less local. One such measure is the **out-degree distribution of the in-neighbors of a given node**. The intuition behind this measure is the in-neighbors of a given page should not simply exist to boost the rank of that page. By plotting these distributions, we can gain insight into the PageRank support of various real web pages. Computation of these degrees was also done with a MapReduce by joining the `PTable<int,int>` mapping nodes to out-degrees with a `PTable<int, int>` corresponding the node ids themselves. This allows for easy filtering of non-neighboring nodes. The parallel nature of this computation allowed these values to be computed in between 50 and 90 seconds. Plots were generated for all top 100 pages by PageRank and are available in the `zip` file submitted along with this paper.

Most of the plots looked like Figures 5 or 6. These look like either a *power law* or a *lognormal* distribution. To determine which distribution was the case, the `distribution_compare` function in the `powerlaw` library. In 96 cases, the lognormal distribution was chosen. Figure 7 show the p-values determined by the distribution comparison. The low p-values for 60% of the nodes indicate our sureness the the distribution of neighboring nodes is generally a lognormal distribution.

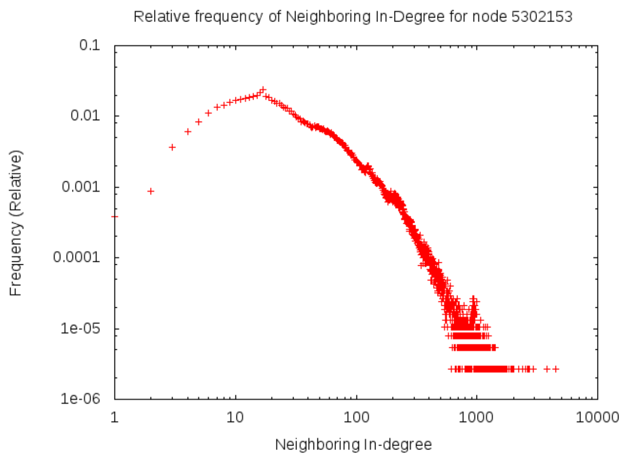

Figure 5: Neighboring Out-Degree distribution for "United States"

All of the top 100 nodes were fit to lognormal distributions. Figures 8, 9, and 10 show the shape, location, and scale values of fitted distributions. The location values (which decide on the left / right shift of the distribution) are clustered around 0, with a range from $[-13, 12]$. All but two shape values were

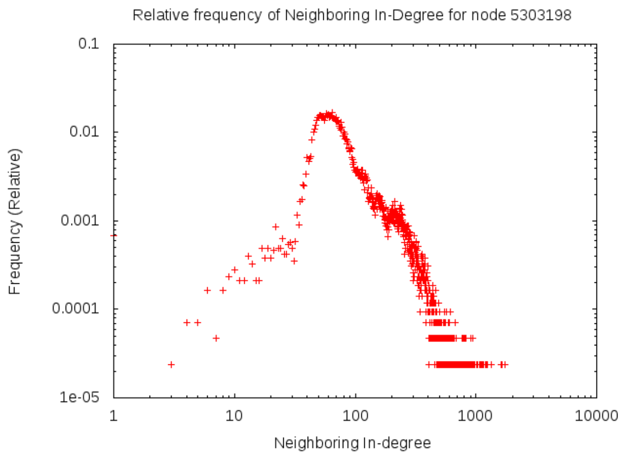

Figure 6: Neighboring Out-Degree distribution for "United States Census Bureau"

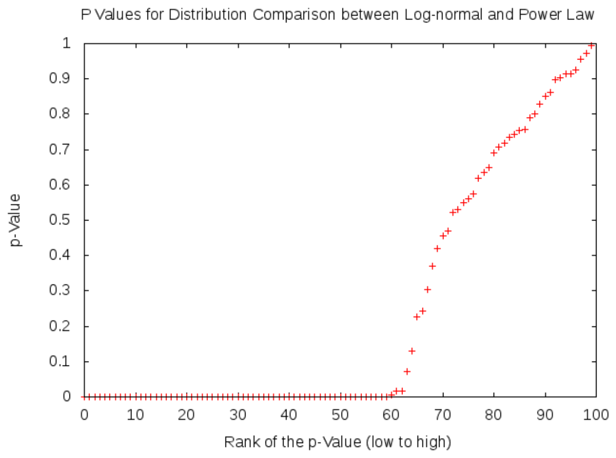

Figure 7: p-values for distribution comparison between lognormal and power law distributions

in the range $(0.55, 1.5)$ and all but one was in the range $(0.55, 2)$. The low outlier was a value of 0.299 and corresponds Figure 11. In particular, the neighboring out-degree distribution of this node looks like the degree distribution of some of our example spam pages as almost all nodes have a degree of 2. In fact, looking at this page (United States Postal Abbreviations), it appears as though many of the degree 2 pages were for small towns pointing at the page due to their state code. Lastly, all but one scale value was greater than 15 (which was for the postal abbreviations page), which had a scale value of close to 1.

This leads us to the following detection mechanism:

1. Compute the out-degree distribution of in-degree of all nodes.
2. Fit the data to a lognormal distribution, obtaining the shape, location, and scale values. If the distribution fits any other distribution (excluding a power law) better than the lognormal, reject the page as spam.
3. If the shape value is outside of the range $(0.55, 1.5)$ or the scale value is less than 15, then consider the page as spam. Otherwise, accept the page as valid content.

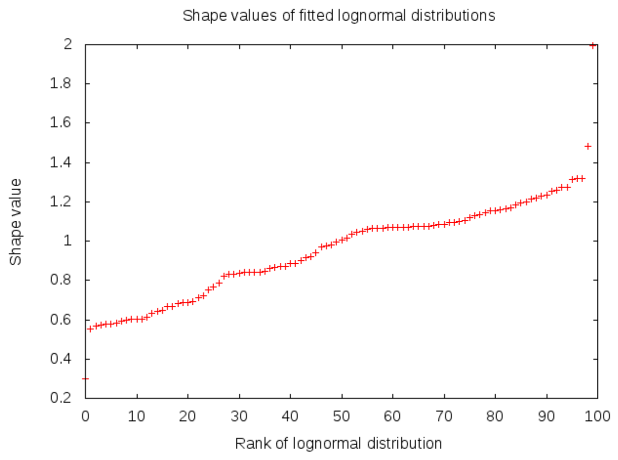

Figure 8: Shape values of the lognormal fits for the top 100 PageRank pages.

Evaluation

From Figures 8, 9, and 10, its clear that 99 of the top 100 nodes by PageRank pass this classification test. Moreover, the single rejected node has hallmarks of spam, meaning that it's fair to state that all 100 top nodes were classified correctly. If we look at the distributions for four types of spam networks, the out degree distributions have nearly 100% probability mass at degree 1 nodes (except for the k target pages, which have 100% mass at degree k nodes) and, as a result, the scale values are all identically 1, as are the shape values. As such, the spam networks fail this criteria.

One method for a spammer to trick the model would be to generate a lognormal distribution from a given

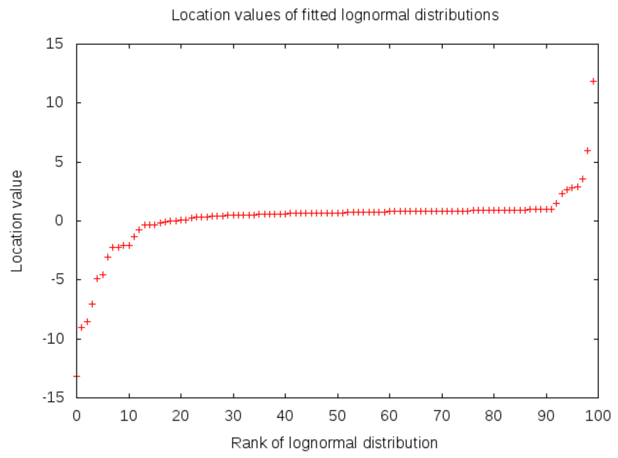

Figure 9: Location values of the lognormal fits for the top 100 PageRank pages.

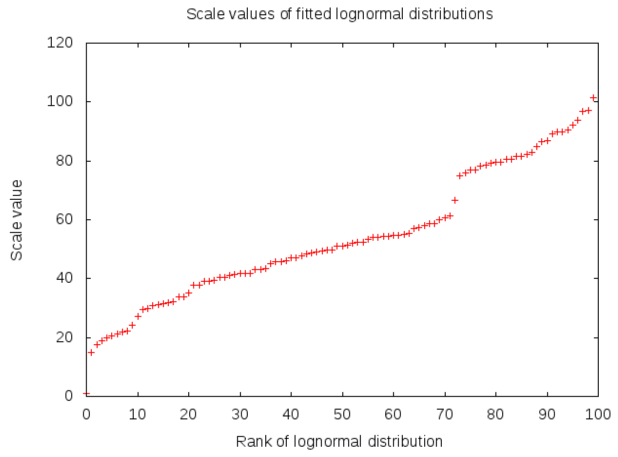

Figure 10: Scale values of the lognormal fits for the top 100 PageRank pages.

shape, location, and scale (that all pass the filter listed above), sample frequencies from this distribution, and then make a graph with a single target page and the desired out-degree distribution of in-neighbors. Note that several samples must be taken in order to cause the data to be fitted back to a lognormal with the appropriate scale and shape.

To show that this is expensive for the spammer, recall that the amount of PageRank inside of a closed system is constant. Thus, we can proxy the PageRank of our target page by looking at the relative PageRank of the target page versus the spam pages.

The raw data for the 100 top pages was used to create

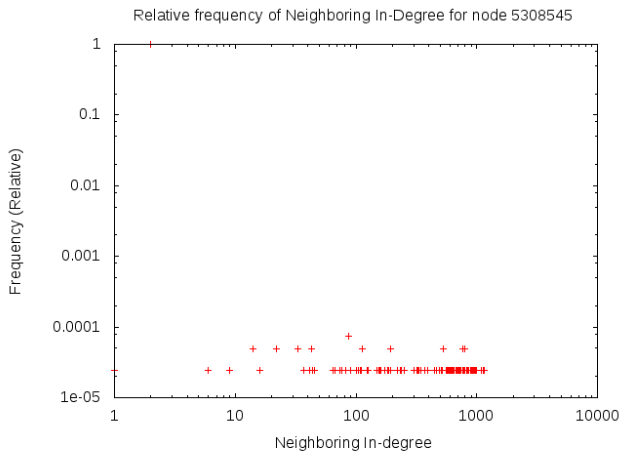

Figure 11: Neighboring Out-Degree Distribution for the US Postal Abbreviations.

100 graphs, each with a single target nodes and an out-degree distribution of in-neighbors that matched the raw data. To maximize the PageRank of the target page, lower id nodes were preferred for destinations of edges (in order to "funnel" page rank toward to target page as in the case of the graph with intermediate nodes).

Then, a PageRank calculation was run on this network, computing the relative portion of page rank in the target page versus the rest of the network. In the single target, reciprocated graph, $\approx 46\%$ of the PageRank goes to the target page. However, in these networks, between 15% and 27% of the total PageRank went to the target page. Thus, if there are k boosting pages and p is the fraction of the PageRank that goes to the target page in this network, the amount of actual page rank given to the target page in the actual network is given by $\frac{pk(1-\beta)}{N}$. As the only term that varies between, the ideal spam graph and this new graph is p, the spammer must add nodes proportional to the ratios of the scores p, which results in an increase in the number of boosting page size of between $1.7x$ and $3x$ versus the ideal spam, meaning that the spammer must now create as many in-edges as a real page would have.

The most exciting part of this comes from noting that this algorithm works nicely with TrustRank and Spam Mass Estimation[3]. In that method, spammers would need a sufficiently large mass of PageRank to emanate from the good core of pages. Once this threshold was passed, they could supplement this

PageRank with relatively few pages in order to reach their target PageRank (as the PageRank boost from the trusted pages might be quite large). However, if we apply this algorithm on top of Spam Mass Estimation, the spammer must now create several additional spam pages in order to satisfy the out-degree distribution requirement from this check. As such, the spammer is forced to create these boosting pages wastefully, though their PageRank contributions might not be needed in order to change the position of the page.

Conclusion

In this paper, we presented a method of identifying link spam on the web based on local graph properties. We used empirical properties about real life web pages with high PageRank to show that the out-degree of in-neighbors fit a lognormal distribution and that this fit is a better fit than a power law or other standard distributions. We specified a decision algorithm based on the shape and scale parameters of the lognormal distribution to classify pages as spam or not spam. We found that this classification method detects common link spam graphs and, for a spammer to trick the classifier, they must create a graph with roughly the same amount of edges and nodes as a neighborhood of size 2 around the real Wikipedia nodes. The resulting PageRank is comparable to the Wikipedia page nodes, resulting in a 1.7x - 3x increase in the cost to the spammer. When coupled with Spam Mass Estimation, this forces spammer to hijack links from the trusted part of the web and also generate huge number of boosting pages, many of which will not be allowed to contribute significantly to the PageRank of the target page.

Future Work

All of the data used in this page is from a 5 year old snapshot of Wikipedia. It might be interesting to see if this distribution can be found throughout the web or if it is just an artifact of Wikipedia's link structure.

It will also be worht investigating the minimum number of pages needed for a spammer to fool the classifier during lognormal fit step. If an efficient mechanism can be found to do this with only a few pages, then this method will be rendered useless.

In terms of increasing the challenge for the spammer, it might also be worth looking one level sideways in terms of the out-degree distribution (i.e., the "out-degree distribution of the in-neighbors" for the out-neighbors of the in-neighbors of a given node). The intuition here is that is the lognormal distribution is found to be a property of all pages (and not just the high PageRank pages), then spammers must construct networks with so many constraints on degrees and edges that their problem becomes computationally intractable. This will also mitigate any spammer success in solving the previous open question, as the complexity of the mechanism increases.

References

[1] Using the wikipedia page-to-page link database. `http://haselgrove.id.au/wikipedia.htm`. Accessed: 2014-10-31.

[2] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107–117, April 1998.

[3] Zoltan Gyöngyi, Pavel Berkhin, Hector Garcia-Molina, and Jan Pedersen. Link spam detection based on mass estimation. In *Proceedings of the 32Nd International Conference on Very Large Data Bases*, VLDB '06, pages 439–450. VLDB Endowment, 2006.

[4] Zoltán Gyöngyi and Hector Garcia-Molina. Link spam alliances. In *Proceedings of the 31st International Conference on Very Large Data Bases*, VLDB '05, pages 517–528. VLDB Endowment, 2005.

[5] Zoltán Gyöngyi, Hector Garcia-Molina, and Jan Pedersen. Combating web spam with trustrank. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, pages 576–587. VLDB Endowment, 2004.

[6] Taher H. Haveliwala. Topic-sensitive pagerank. In *Proceedings of the 11th International Conference on World Wide Web*, WWW '02, pages 517–526, New York, NY, USA, 2002. ACM.