SANDIA REPORT

SAND2009-6670
Unlimited Release
Printed October 2009

Generalized BadRank with Graduated Trust

Tamara G. Kolda and Michael J. Procopio

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94-AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2009-6670
Unlimited Release
Printed October 2009

Generalized BadRank with Graduated Trust

Tamara G. Kolda
Informatics and Decision Sciences Department
Sandia National Laboratories
Livermore, CA 94551-9159
Email: tgkolda@sandia.gov

Michael J. Procopio
Next Generation Monitoring Systems Department
Sandia National Laboratories
Albuquerque, NM 87123-0401
Email: mjproco@sandia.gov

Abstract

BadRank is a method for detecting spam web sites, based on the premise that a page is spam if it points to another spam page; i.e., the BadRank score of a page is the weighted sum of the BadRank scores of the pages that it links to. BadRank is an important tool in spam detection. We consider the mathematical structure of BadRank, showing how it can be modified to guarantee that the iterates converge. Additionally, we consider methods for incorporating knowledge about trusted (known non-spam) sites into the BadRank calculation by changing the underlying iteration matrix. The effectiveness of BadRank in web spam detection is demonstrated in a statistically significant evaluation on the WEBSPAM-UK2007 data set.

This page intentionally left blank.

Contents

Figures

Tables

This page intentionally left blank.

1 Introduction

BadRank [22] is a method for detecting spam web sites, based on the premise that a page is spam if it points to another spam page; i.e., the BadRank score of a page is the weighted sum of the BadRank scores of the pages that it links to. BadRank can be considered a random walk that traverses links in reverse and periodically jumps back to a starting seed set of known "bad" pages (similar to Personalized PageRank [20]). This focuses BadRank on pages that are closely linked to a predefined set of known bad pages. Because of this property, BadRank is an important tool in spam detection.

The identification of web spam has been identified as a major challenge for web search engines [13]. Spam web sites deliberately manipulate their placement (or that of paying customers) in search engine rankings. Since web pages that rank highly attract significantly more attention from users, high placement in search results is critical for online commerce sites. In addition to legitimate commerce sites (that may use illegitimate means to be highly ranked), many spam web sites exist simply to serve online advertisements that generate revenue whenever they are clicked. For a recent and detailed assessment of web spam, see [23].

One of the techniques used by spammers is so-called link spam, where farms of interlinked web sites are used to give high PageRank to certain pages. These link farms tend not to have any legitimate content and so do not have incoming links from sites outside the farm. Therefore, if one page within a link farm can be identified, we can reasonably suspect that any pages that point to it are also spam pages. Likewise, pages that point to those pages are likely to be spam, and so on.

In this paper, we generalize the formulation of BadRank and make the following contributions:

- We determine what modifications are required to ensure that the BadRank iteration matrix is stochastic, aperiodic, and irreducible, guaranteeing that the iterates converge. This involves adjusting the hyperlink matrix to account for leaf nodes (nodes with no inlinks) and adding a uniform random jump.
- We develop two novel methods for incorporating trust. The first declares a site as absolutely trusted and removes all its outbound links so that it cannot receive any BadRank score. The second is more flexible and defines a notion of graduated trust. In this case, nodes are weighted according to how trustworthy they might be. This results in non-uniform transition probabilities in the random walk.
- We test 48 different variations of BadRank on the WEBSPAM-UK2007 dataset. By itself, BadRank is not an effective classifier because it can only find spam pages that are connected to those pages in its seed set. We show, however, that it can increase the performance of a classifier when combined with other features.

The paper is organized as follows. Section 2 analyzes the BadRank method and presents novel variations that ensure that the iterates converge and furthermore incorporate trust. Section 3 gives results of BadRank on a real-world dataset that has over 6,000 hand-labeled spam and non-spam web sites. Section 4 surveys related work. Finally, Section 5 discusses conclusions and future work.

Notation

Matrices are denoted by boldface capital letters ($\mathbf{A}, \mathbf{B}, \mathbf{C}, \dots$) and vectors by boldface lower case letters ($\mathbf{a}, \mathbf{b}, \mathbf{c}, \dots$). Indices are denotes by lowercase letters ($i, j, k, \dots$). The (i, j) element of a matrix $\mathbf{A}$ is denoted by $\mathbf{A}(i, j)$, and the ith element of a vector $\mathbf{a}$ is denoted by $\mathbf{a}(i)$. Sets are denoted by calligraphic fonts ($\mathcal{A}, \mathcal{B}, \mathcal{C}, \dots$). Sizes of sets or numbers of items are denotes by uppercase letters ($N, M, \dots$). We let $\mathbf{e}$ denote the vector of all ones and $\mathbf{I}$ denote the identity matrix. The *one-norm* of a vector is defined as

$$\| \mathbf{x} \|_1 = \sum_i |\mathbf{x}(i)|.$$

We let $\mathbb{R}$ denote the set of all real numbers, $\mathbb{R}_{\geq 0} = [0, +\infty)$ denote the set of all nonnegative numbers, and $\mathbb{R}_+ = (0, +\infty)$ denote the set of strictly positive real numbers. A nonnegative square matrix is called *stochastic* if its rows sum to one; i.e., $\mathbf{A} \in \mathbb{R}_{\geq 0}^{N \times N}$ is stochastic if

$$\sum_j \mathbf{A}(i, j) = 1 \text{ for } i = 1, \ldots, N.$$

2 The BadRank Method

We consider links between web pages or web hosts. For ease of exposition, we discuss all methods in terms of page-to-page links, but in practice we will consider a host-to-host graph. The mathematics is equivalent.

We assume that we have a collection of N web pages and a seed set of known "bad" pages denoted by $\mathcal{B} \subseteq \{1, \ldots, N\}$. Let $M = |\mathcal{B}|$ denote the number of bad pages. A hyperlink in the web graph from page i to page j is denoted by $i \to j$. We ignore self-links in the data and only count each hyperlink only once, even if there are multiple links between two pages. The $N \times N$ hyperlink matrix is given by

$$\mathbf{H}(i,j) = \begin{cases} 1 & \text{if } i \to j \\ 0 & \text{otherwise} \end{cases} \quad \text{for} \quad 1 \le i, j \le N. \tag{1}$$

2.1 Basic BadRank

The BadRank method has not been formally published, but is discussed on a web page that speculates on Google's spam detection methods [22]. A closely related method is Anti-Trust Rank [15]. In BadRank, we compute a sequence

$$\mathbf{s}_k(i) = \alpha \sum_{i \to j} \frac{\mathbf{s}_{k-1}(j)}{\mathbf{inlinks}(j)} + (1-\alpha)\mathbf{b}(i) \qquad \text{for} \quad 1 \le i \le N,$$

where $\mathbf{s}_k \in \mathbb{R}^N_{\ge 0}$, $\mathbf{s}_k(i)$ is the *BadRank score* of page i at iteration k, $\mathbf{inlinks}(i)$ is the number of inlinks to page i, and α is some parameter in the interval $[0, 1]$ (typically $\alpha = 0.85$). In [22], the vector $\mathbf{b} \in \mathbb{R}^N_{\ge 0}$ is not completely specified but instead described as a "special evaluation of certain web pages ... [that] reflects if a page was detected by a spam filter or not."[1] For our purposes, we assume

$$\mathbf{b}(i) = \begin{cases} 1/M & \text{if } i \in \mathcal{B}, \\ 0 & \text{otherwise} \end{cases} \quad \text{for} \quad 1 \le i \le N. \tag{2}$$

Typically, the iterates are initialized by $\mathbf{s}_0 = \mathbf{b}$.

BadRank is identical to Personalized PageRank [20, Section 6] with the personalization vector set to $\mathbf{b}$, except that the links are reversed. In fact, BadRank can be considered a variant of Topic-Sensitive PageRank [12] and TrustRank [11]. If $\alpha = 0.85$, the random surfer will walk backwards over *inbound* links 85% of the time and then jump to a random page in $\mathcal{B}$ the other 15% of the time. Another way to think of this is that the random surfer will, on average, jump back to a random page in $\mathcal{B}$ approximately every six steps. The main problem with personalization is that the resulting iteration matrix may not be aperiodic and irreducible, which we discuss the implications of in §2.2.

BadRank can be formulated in matrix notation as

$$\mathbf{s}_k^\mathsf{T} = \mathbf{s}_{k-1}^\mathsf{T} \mathbf{B},$$

where $\mathbf{B} \in \mathbb{R}^{N \times N}_{\ge 0}$ is the *BadRank matrix* defined as

$$\mathbf{B} = \alpha \mathbf{D}\mathbf{H}^\mathsf{T} + (1-\alpha)\mathbf{e}\mathbf{b}^\mathsf{T} \tag{3}$$

[1] Reference [22] continues on to say that the sum of $\mathbf{b}$ should be equal to N, but a more accurate statement would be to say that it should be equal to $\| \mathbf{s}_0 \|_1$. If the initial set of scores is all ones, then $\mathbf{b}$ should sum to N; likewise, if the initial set of scores sums to one, then $\mathbf{b}$ should also sum to one.

and $\mathbf{D} \in \mathbb{R}_{\geq 0}^{N \times N}$ is a diagonal scaling matrix whose ith diagonal entry is given by

$$\mathbf{D}(i,i) = \begin{cases} 1/\mathbf{inlinks}(i) & \text{if } \mathbf{inlinks}(i) > 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{for} \quad 1 \leq i \leq N. \quad (4)$$

If every page has at least one inlink, then $\mathbf{B}$ is stochastic; otherwise, it is substochastic. We discuss the implications of this in §2.2.

Krishnan and Raj [15] show BadRank (which they refer to as Anti-Trust Rank) is effective in detecting spam pages.

2.2 Generalized BadRank Formulation

In this subsection, we present a generalized formulation of the BadRank matrix and consider the BadRank method of [22, 15] as a special case. Consider the Badrank iteration:

$$\mathbf{s}_k^\mathsf{T} = \mathbf{s}_{k-1}^\mathsf{T}\mathbf{B} = \mathbf{s}_0^\mathsf{T}\mathbf{B}^k \quad \text{for} \quad k = 1, 2, \ldots. \quad (5)$$

It is well known that if $\mathbf{B}$ is stochastic, irreducible, and aperiodic, then a unique stationary distribution of the Markov Chain defined by $\mathbf{B}$ exists which satisfies

$$\mathbf{s}^\mathsf{T} = \mathbf{s}^\mathsf{T}\mathbf{B}.$$

Moreover, the iteration defined in (5) is guaranteed to converge to the unique stationary distribution regardless of the choice for $\mathbf{s}_0$. If $\mathbf{B}$ is substochastic, then the iterates defined by (5) may converge to zero. If $\mathbf{B}$ is not aperiodic and irreducible, then the iterates defined by (5) may cycle or converge to different solutions depending on the starting guess. See, e.g., [16] for a discussion. Oftentimes these problems are not observed in practice because the number of iterations is fixed to some low value such as 25 and different starting points are not evaluated.

We define a generalized form of the BadRank matrix, $\mathbf{B} \in \mathbb{R}_{\geq 0}^{N \times N}$, as

$$\mathbf{B} = \alpha\mathbf{W} + \beta\mathbf{e}\mathbf{b}^\mathsf{T} + \gamma\mathbf{e}\mathbf{v}^\mathsf{T}. \quad (6)$$

We assume $\alpha + \beta + \gamma = 1$ as well as the following.

- The matrix $\mathbf{W} \in \mathbb{R}_{\geq 0}^{N \times N}$, called the *walk matrix*, is assumed to be at least substochastic. It controls the probability of walking backwards along existing links. A walk step is followed with the probability specified by α.
- The vector $\mathbf{b} \in \mathbb{R}_{\geq 0}^{N}$ is such that $\| \mathbf{b} \|_1 = 1$ and is called the *known bad node jump vector*. It is used whenever we jump to a known bad node and determines the probability distribution among the nodes in $\mathcal{B}$. We assume throughout that $\mathbf{b}$ is as defined in (2) so that every bad node is equally probable. Further, we assume that many entries in $\mathbf{b}$ are zero, corresponding to nodes outside of $\mathcal{B}$. A known bad node jump is taken with the probability specified by β.
- The vector $\mathbf{v} \in \mathbb{R}_{+}^{N}$ is such that $\| \mathbf{v} \|_1 = 1$ and is called the *random node jump vector*. It controls the probabilities of jumping to a completely random node. We assume throughout that $\mathbf{v} = \mathbf{e}/N$ so that every node is equally probable. It is possible to choose any probabilities so long as they are all strictly positive and sum to one. A completely random jump is taken with the probability specified by γ.

Theorem 2.1 *In (6), if* $\mathbf{W}$ *is stochastic, then* $\mathbf{B}$ *is stochastic. Further, if* $\gamma > 0$*, then* $\mathbf{B}$ *is aperiodic and irreducible.*

The proof is straightforward and so is omitted; see [16].

The basic BadRank formulation in §2.1 has

$$\beta = 1 - \alpha \quad \text{and} \quad \gamma = 0.$$

Consequently, the third term in (6) is not present in this formulation. This is a potential pitfall since the resulting stochastic matrix may be reducible or aperiodic, meaning that the iterations defined by (5) may not converge, as mentioned above. A more serious problem has to do with the walk matrix. In basic BadRank, we have

$$\mathbf{W} = \mathbf{D}\mathbf{H}^{\mathsf{T}}$$

where $\mathbf{D}$ is as defined in (4) and $\mathbf{H}$ as in (1). We refer to pages without any inlinks as *leaf nodes*. If any leaf nodes exist, then $\mathbf{W}$ is substochastic. This means that the scores defined in (5) may converge to zero.

Consider the example shown in Figure 1. We assume $\mathcal{B} = \{1\}$. The BadRank scores calculated by the basic BadRank formulation are given in Table 1. The values change depending on the number of iterations and eventually converge to zero. We consider a straightforward method to correct for this problem in the next subsection.

2.3 Ensuring Stochasticity of the BadRank Matrix

As mentioned in the previous subsection, the basic BadRank matrix may be substochastic due to the presence of leaf nodes. We define the set of leaf nodes as

$$\mathcal{L} = \{i : \mathbf{inlinks}(i) = 0\}.$$

In the matrix $\mathbf{H}$, leaf nodes correspond to zero columns. One possible fix is to remove any leaf nodes (the original PageRank paper [20] proposes such an approach), but the removal of leaf nodes may produce yet more leaf nodes, requiring an iterative removal process. Moreover, we will not have BadRank scores for the leaf nodes (though these can be computed; see, e.g., [14]). Another more attractive possibility is to add artificial links pointing to the leaf nodes. We propose three options for creating a modified hyperlink matrix $\hat{\mathbf{H}} \in \mathbb{R}_{\geq 0}^{N \times N}$ which has no zero columns. We can then define the stochastic matrix $\mathbf{W}$ in (6) as

$$\mathbf{W} = \hat{\mathbf{D}}\hat{\mathbf{H}}^{\mathsf{T}} \tag{7}$$

where $\hat{\mathbf{D}} \in \mathbb{R}_{\geq 0}^{N \times N}$ is a diagonal scaling matrix defined by

$$\hat{\mathbf{D}}(i,i) = 1/\sum_{j=1}^{N} \hat{\mathbf{H}}(i,j). \tag{8}$$

Leaf Self Links

One idea is to add self links for all the leaf nodes. We define a modified hyperlink matrix as

$$\hat{\mathbf{H}} = \mathbf{H} + \mathbf{L}$$

where $\mathbf{L} \in \mathbb{R}^{N \times N}$ is a diagonal matrix such that

$$\mathbf{L}(i,i) = \begin{cases} 1 & \text{if } i \in \mathcal{L} \\ 0 & \text{otherwise} \end{cases} \quad \text{for } i = 1, \ldots, N.$$

Leaf Bad Links

Another idea is to jump back to the known bad nodes whenever we encounter a leaf node. This add links from the bad nodes pointing to the leaf node. Therefore, we define

$$\hat{\mathbf{H}} = \mathbf{H} + \mathbf{J}$$

where $\mathbf{J} \in \mathbb{R}_{\geq 0}^{N \times N}$ is a matrix such that

$$\mathbf{J}(i,j) = \begin{cases} 1 & \text{if } i \in \mathcal{B} \text{ and } j \in \mathcal{L} \\ \mathbf{0} & \text{otherwise} \end{cases} \quad \text{for } i = 1, \ldots, N.$$

In other words, we fill in the empty columns of $\mathbf{H}$ with copies of the vector $M\mathbf{b}$.

Self Links

Another option is to just add self links for every node. In this case, we have no need to predetermine the leaf nodes. This may be the most appropriate option in a streaming environment. In this case,

$$\hat{\mathbf{H}} = \mathbf{H} + \mathbf{I}$$

where $\mathbf{I}$ is the $N \times N$ identity matrix.

Examples of fixing the leaf nodes

Consider again the hyperlink graph in Figure 1. We experiment with each of the three fixes listed above by running BadRank until $\|s_k - s_{k-1}\| \leq \epsilon$, where $\epsilon = 10^{-10}$; this required feweer than 100 iterations in all three examples. The results are given in Table 2. Here we can see that adding "leaf self links" or "self links" results in a large score for the leaf node. This is because it becomes a sink node. We would expect that Node 2 should have a high BadRank score because it points directly to the known bad node. Likewise, nodes 3 and 4 are two hops from the known bad node, and Node 5 is three hops away, and we would expect the scores to descend accordingly. We see the expected behavior for "leaf bad links." Interestingly, "self links" does best in our experiment results on real-world data shown in §3.

2.4 Incorporating Trust

In many cases, we may have a known set of trusted nodes, denoted by the set $\mathcal{T}$.[2] For example, we may want to specify that search engines such as Google and Yahoo! are trusted sites even though they have links to known spam sites. Additionally, we may want to trust sites that have already been manually inspected and determined not to be spam.

Consider the example shown in Figure 2. Here, $\mathcal{B} = \{1, 2\}$ and $\mathcal{T} = \{3\}$. BadRank scores are given in the second column of Table 3. We see that Node 3 gets a high BadRank score; moreover, Node 6 also has a high BadRank score even though it only has one link and that to a known trusted node. Even if we were to postprocess the scores in order to set Node 3's BadRank score to zero, this would not correct the BadRank score of Node 6. Therefore, we propose to consider a method that incorporates trust directly into the BadRank computation.

[2] We naturally require $\mathcal{T} \cap \mathcal{B} = \emptyset$.

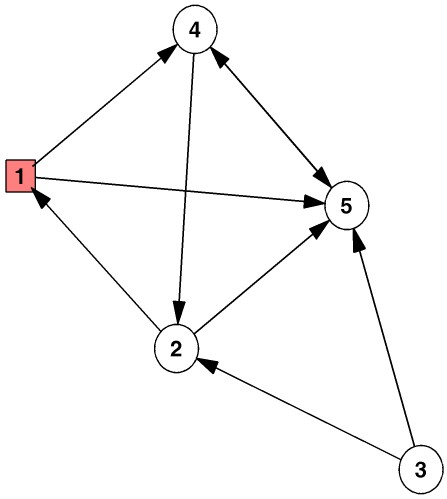

Figure 1. Example hyperlink graph. Node 1 is drawn with a red rectangle to indicate that it is a known bad node. Node 3 is a leaf node because it has no inlinks.

Table 1. Basic BadRank ($\alpha = 0.85, \beta = 0.15, \gamma = 0$) scores for Figure 1. The iterates converge to zero due to the presence of leaf nodes.

	Iteration			
Node	15	30	45	60
1 (bad)	0.0330	0.0032	0.0003	0.0000
2	0.0350	0.0034	0.0003	0.0000
3	0.0198	0.0019	0.0002	0.0000
4	0.0198	0.0019	0.0002	0.0000
5	0.0099	0.0010	0.0001	0.0000

Table 2. BadRank ($\alpha = 0.84, \beta = 0.15, \gamma = 0.01$) scores for Figure 1 with different choices for $\mathbf{W}$ according to how the leaf nodes are fixed.

	Fix		
Node	leaf self links	leaf bad links	self links
1 (bad)	0.1942	0.3457	0.3119
2	0.1728	0.3054	0.1919
3 (leaf)	0.5141	0.1433	0.3807
4	0.0823	0.1433	0.0846
5	0.0366	0.0622	0.0309

Our goal is to remove all outbound hyperlinks from trusted nodes so that they cannot receive any BadRank score. We define an anti-trust vector $\mathbf{z} \in \mathbb{R}^N_{\geq 0}$ so that trusted sites has a value of zero, i.e.,

$$\mathbf{z}(i) = \begin{cases} 0 & \text{if } i \in \mathcal{T} \\ 1 & \text{otherwise.} \end{cases} \tag{9}$$

In the simplest case, we define

$$\hat{\mathbf{H}} = \mathbf{ZH}$$

where $\mathbf{Z} = \text{diag}(\mathbf{z})$, which defines a modified hyperlink matrix with all links from trusted nodes removed. The matrix $\mathbf{W}$ can then be defined as usual in (7) except that we have not corrected for leaf nodes. (Note the equation for $\hat{\mathbf{D}}$ in (8) can be trivially modified to account for the possible zero columns in $\hat{\mathbf{H}}$.)

We reconsider the three methods for fixing leaf nodes in the presence of trusted nodes below. Before we proceed with this discussion, we note that removing some links may create additional leaf nodes. Therefore, we define an amended set of leaf nodes, $\bar{\mathcal{L}}$, to be the set of those nodes that have no inlinks once outbound links from trusted nodes have been removed, i.e.,

$$\bar{\mathcal{L}} = \{j \mid \sum_{i=1}^{N} \mathbf{z}(i)\mathbf{H}(i,j) = 0\}.$$

It must be the case that $\mathcal{L} \subseteq \bar{\mathcal{L}}$. In Figure 2, we have $\mathcal{L} = \{6, 10\}$ and $\bar{\mathcal{L}} = \{2, 4, 6, 10\}$.

Leaf Self Links with Trust

The only modifications in the case of adding links of leaf nodes is to use the amended set of leaf nodes to form $\mathbf{L}$. We define a modified hyperlink matrix as

$$\hat{\mathbf{H}} = \mathbf{ZH} + \bar{\mathbf{L}}$$

where $\bar{\mathbf{L}} \in \mathbb{R}^{N \times N}$ is a diagonal matrix such that

$$\bar{\mathbf{L}}(i,i) = \begin{cases} 1 & \text{if } i \in \bar{\mathcal{L}} \\ 0 & \text{otherwise} \end{cases} \quad \text{for } i = 1, \ldots, N. \tag{10}$$

Leaf Bad Links with Trust

In the case of adding leaps back to bad nodes, we weight the jump by the appropriate entry of $\mathbf{z}$ and once again use the amended set of leaf nodes. Therefore, we define

$$\hat{\mathbf{H}} = \mathbf{ZH} + \mathbf{J}$$

where $\mathbf{J} \in \mathbb{R}^{N \times N}_{\geq 0}$ is a matrix such that

$$\mathbf{J}(i,j) = \begin{cases} \mathbf{z}(i) & \text{if } i \in \mathcal{B} \text{ and } j \in \bar{\mathcal{L}} \\ \mathbf{0} & \text{otherwise} \end{cases} \quad \text{for } i = 1, \ldots, N.$$

Self Links with Trust

Another option is to just add self links for every node. This may be the most appropriate option in a streaming environment. In this case,

$$\hat{\mathbf{H}} = \mathbf{Z}(\mathbf{H} + \mathbf{I}) + \tilde{\mathbf{L}}$$

where $\mathbf{I}$ is the $N \times N$ identity matrix and $\tilde{\mathbf{L}}$ is as

$$\tilde{\mathbf{L}}(i,i) = \begin{cases} 1 & \text{if } i \in \bar{\mathcal{L}} \cap \mathcal{T} \\ 0 & \text{otherwise} \end{cases} \quad \text{for } i = 1, \ldots, N. \tag{11}$$

Example

We experiment by adding trust into the computation of BadRank for Figure 2. We run BadRank until $\|s_k - s_{k-1}\| \leq \epsilon$, where $\epsilon = 10^{-10}$; this required fewer than 100 iterations in both experiments. The BadRank scores with trust are shown in the third column of Table 3. In contrast to the results without trust in column 2, the BadRank score of Node 3 is now very small (it is not exactly zero because of the random jumps produced by positive values of γ). Further, the BadRank score of Node 6 is also very small because there was no unwanted propagation of the BadRank score through Node 3.

2.5 Incorporating Graduated Trust

Lastly, we consider the case where some nodes have a degree of trustworthiness that is not absolute. We define an anti-trust vector $\mathbf{z}$ more generally than in (9). For example, we may have TrustRank scores from an alternate calculation. In this case, we allow for pages in which we have some trust, i.e.,

$$0 < \mathbf{z}(i) < 1 \text{ if node } i \text{ is somewhat trusted.}$$

We still assume $\mathbf{z}(i) = 0$ for all $i \in \mathcal{T}$ and $\mathbf{z}(i) = 1$ for all $i \in \mathcal{B}$. Lower scores indicate higher trust. The formulas from §2.4 are unchanged with this more general definition of the anti-trust vector, $\mathbf{z}$.

Consider Figure 1 and partially trusting one node. We set $\mathbf{z}(i) = 1$ for all nodes except the partially trusted node, for which $\mathbf{z}(i) = 0.10$. Table 4 shows the results.

The effect of graduated trust is that it results in non-uniform transition probabilities in the random walk. Consider the case where Node 4 is partially trusted so that

$$\mathbf{z} = \begin{bmatrix} 1 & 1 & 1 & 0.1 & 1 \end{bmatrix}^{\mathsf{T}}.$$

Then the walk matrix $\mathbf{W}$ is given by

$$\mathbf{W} = \begin{bmatrix} 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.91 & 0.09 & 0.00 \\ 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.50 & 0.00 & 0.00 & 0.00 & 0.50 \\ 0.32 & 0.32 & 0.32 & 0.03 & 0.00 \end{bmatrix}$$

In this case, even though Node 4 links to Nodes 2 and 5, the probability of making a transition from those nodes to Node 4 in the random walk is greatly reduced. For example, Node 2 is 91% likely to step to Node 3 and only 9% likely to step to Node 4. Without the graduated trust, the probabilities would have been 50% for each.

Observe the case where Node 2 is partially trusted—it has almost no effect on the final scores. This is because Node 2 is the only node pointing to Node 1. Consequently, there is a 100% chance that the step will lead to Node 2 is a the walker takes a step according to $\mathbf{W}$. A topic of future research is considering whether more extreme measures should be taken to deflect BadRank score from nodes with graduated trust.

2.6 Algorithm

The BadRank algorithm, incorporating fixes for leaf nodes and trust, is now given. We assume that the hyperlink matrix $\mathbf{H}$ and the set of known bad nodes $\mathcal{B}$, along with the sizes N and M, are given. We

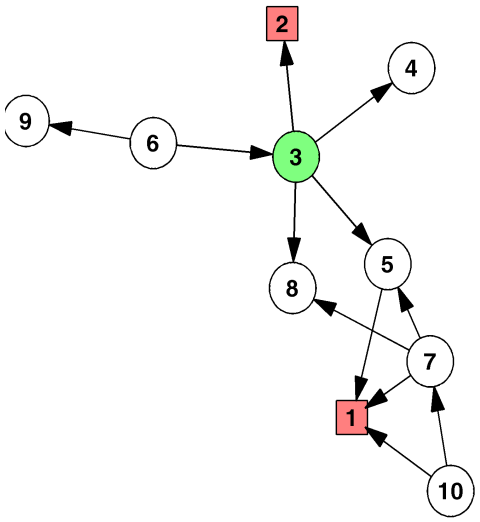

Figure 2. Another example hyperlink graph. Nodes 1 and 2 are drawn with a red rectangle to they are known bad nodes. Node 3 is colored green to indicate it is a known trusted node.

Table 3. BadRank scores ($\alpha = 0.84, \beta = 0.15, \gamma = 0.01$, fix = 'leaf bad links') for Figure 2 with and without trust.

	Trust	
Node	none	binary
1 (bad)	0.1949	0.2812
2 (bad)	0.1949	0.2812
3 (good)	0.1893	0.0010
4	0.0010	0.0010
5	0.0556	0.0797
6	0.1609	0.0027
7	0.0793	0.1475
8	0.0010	0.0010
9	0.0010	0.0010
10	0.1222	0.2037

Table 4. BadRank scores ($\alpha = 0.84, \beta = 0.15, \gamma = 0.01$, fix = 'leaf bad links') with graduated trust results for Figure 1.

	Partially Trusted Node			
Node	2	3	4	5
1	0.3507	0.3124	0.3803	0.3808
2	0.2983	0.2941	0.3251	0.3245
3	0.1442	0.0274	0.2539	0.1410
4	0.1442	0.2563	0.0272	0.1410
5	0.0626	0.1097	0.0134	0.0128

further assume that the anti-trust vector $\mathbf{z}$ has been defined (if trust is not being used, set $\mathbf{z}(i) = 1$ for all $i = 1, \ldots, N$), and that $\mathbf{Z} = \text{diag}(\mathbf{z})$. The parameters α, β, and γ are specified so that $\alpha + \beta + \gamma = 1$. We assume $\mathbf{b}$ is as in (2) and $\mathbf{v} = \mathbf{e}/N$. The algorithm proceeds as follows.

$\mathbf{H} \leftarrow \mathbf{ZH}$
if fix = 'self links' **then**
    $\mathbf{H} \leftarrow \mathbf{H} + \mathbf{Z}$
end if
$\mathcal{L} \leftarrow \{j \mid \sum_{i=1}^{N} \mathbf{H}(i,j) = 0\}$
if fix = 'leaf self links' or 'self links' **then**
    for $j \in \mathcal{L}$ **do**
        $\mathbf{H}(j,j) \leftarrow 1$
    end for
else if fix = 'leaf bad links' **then**
    for $j \in \mathcal{L}$ **do**
        for $i \in \mathcal{B}$ **do**
            $\mathbf{H}(i,j) \leftarrow \mathbf{z}(i)$
        end for
    end for
end if
$\mathbf{D} \leftarrow \mathbf{I}$
for $j = 1, \ldots, N$ **do**
    if $\sum_{i=1}^{N} \mathbf{H}(i,j) > 0$ **then**
        $\mathbf{D}(j,j) \leftarrow 1/\sum_{i=1}^{N} \mathbf{H}(i,j)$
    end if
end for
$\mathbf{W} \leftarrow \mathbf{DH}^{\mathsf{T}}$
$\mathbf{s} \leftarrow \mathbf{b}$
repeat
    $\mathbf{s}^{\mathsf{T}} \leftarrow \alpha \mathbf{s}^{\mathsf{T}} \mathbf{W} + \beta \mathbf{b}^{\mathsf{T}} + \gamma \mathbf{v}^{\mathsf{T}}$
until convergence

This page intentionally left blank.

3 Numerical Results on WebSpam Data

The goal of this section is to show that the addition of BadRank as a feature is useful in spam classification. In our experiments, regardless of the parameterization of BadRank selected, its inclusion increases classification performance.

3.1 Data Description

We tested the variations of BadRank mentioned above on the WEBSPAM-UK2007 dataset [25] using the 277 precomputed features available on the web site (see Table 5) and as described in [3, 5]. The hostgraph had a total of 114,529 nodes and 1,836,441 edges. The labeled data on the web site is broken into two sets for training and testing in the original contest. There is a total of 5709 non-spam (negative) nodes and 344 spam (positive) nodes. Some nodes were also labeled as unsure, but those were treated as unlabeled for the purposes of our experiments. We used 5x2 cross-validation [8] to divide the labeled data into ten equal-sized testing and training folds; sampling was done in a class-stratified manner.

Table 5. WebSpam Features

Feature Type	**Number**
Direct	2
Link-Based	41
Link-Based Transformed	138
Content-Based	96

3.2 Experimental Set-up

The BadRank features are dependent on the training and testing split and are computed as follows. For each of the ten folds of the data, we computed BadRank scores according to the following procedure. We set the vector $\mathbf{b}$ as specified in (2) with $\mathcal{B}$ equal to the set of all known spam nodes in the training set. In cases where trust is incorporated, the vector $\mathbf{z}$ defined as in (9) with $\mathcal{T}$ equal to the set of known non-spam nodes in the training set. We calculate BadRank scores with the parameters listed below for all 114,529 nodes in the host graph and extract just those scores for the 6,053 nodes in our training and testing sets.

We ran BadRank until $\|s_k - s_{k-1}\| \leq \epsilon$ for $\epsilon = 10^{-10}$, or 100 iterations (whichever came first), with all combinations of the following parameterizations:

- $\beta \in \{0.10, 0.15, 0.20\}$,
- $\gamma \in \{0, 0.01\}$,
- fix $\in \{\text{none}, \text{leaf self links}, \text{leaf bad links}, \text{self links}\}$,
- trust $\in \{\text{none}, \text{binary}\}$.

This results in a total of 48 different BadRank variations.

Each feature within the training and testing data is separately scaled to the range $[0, 1]$ by an affine linear transformation. For the classifier, we used LIBSVM Version 2.88-1 [6] for MATLAB with the Gaussian (Radial Basis Function) kernel and the parameter gamma set to 0.05. This choice of gamma was determined via preliminary experiments using only the precomputed features. Probabilistic output on $[0, 1]$ was obtained from the raw SVM decision values using Platt's scaling method [21]. The cost parameter c was kept at the default value of 1 throughout the experiments.

Table 6. Mean AUC Scores with BadRank feature.

AUC w/o BadRank = 0.7328		**fix**							
		none		leaf self links		leaf bad links		self links	
		trust		**trust**		**trust**		**trust**	
β	γ	none	binary	none	binary	none	binary	none	binary
0.10	0.00	0.7462	0.7429	0.7454	0.7432	0.7468	0.7431	0.7474	0.7444
0.10	0.01	0.7459	0.7426	0.7452	0.7429	0.7467	0.7432	0.7469	0.7442
0.15	0.00	0.7473	0.7439	0.7471	0.7443	0.7468	0.7438	0.7490	0.7458
0.15	0.01	0.7464	0.7432	0.7470	0.7443	0.7465	0.7432	0.7487	0.7455
0.20	0.00	0.7466	0.7450	0.7484	0.7454	0.7469	0.7456	0.7497	0.7467
0.20	0.01	0.7462	0.7450	0.7483	0.7452	0.7466	0.7453	0.7496	0.7462

3.3 Results

For each of the ten folds of the data, we trained and tested the SVM with the original 277 precomputed features and then did 48 more tests with the 277 original features plus one parameterization of the BadRank feature. We use Area Under the ROC Curve (AUC) as the performance metric because it is robust even when the data are imbalanced (as they are in this case); moreover, AUC was the metric used in the "web spam challange"[3], which was based on the same data used in this study.

The results (averaged across all ten folds) are given in Table 6. Without any BadRank feature, i.e., our baseline approach, the mean AUC score across the 10 cross-validation folds was 0.7328 with a standard deviation of 0.0196. Given that we are only adding one feature to 277 already existing features, we do not expect a large change in the score. Using BadRank with all combinations of parameters mentioned above, and adding only one at a time to the existing set of 277 features, the mean AUCs ranged from 0.7426 to 0.7497 with standard deviations between 0.0177 and 0.0220.

The best-performing BadRank parameterization in terms of the mean AUC scores across all ten folds of data was $\beta = 0.20$, $\gamma = 0$, trust = "none," and fix = "self links." In a matched-pairs t-test, this parameterization is statistically significantly better that the baseline at the 99% confidence level with a p-value of 0.0001.

The worst-performing BadRank parameterization in terms of the mean AUC scores across all ten folds of data was $\beta = 0.05$, $\gamma = 0.01$, trust = "binary," and fix = "none." In a matched-pairs t-test, this parameterization is still statistically significantly better that the baseline at the 99% confidence level with a p-value of 0.001. We draw two key conclusions from the above results. First, for the Web Spam Challenge data set, using BadRank scores as a feature contributes useful information that significantly improves performance in the spam/non-spam binary classification problem. Such a significant performance benefit from the inclusion of a single feature suggests that BadRank is providing useful information that is indpendent of other features, including many link-based features that were computed from the web and/or host graph.

Second, although we have identified a number of parameters that influence the operation of BadRank, even the worst-performing parameterization yielded statistically significantly improved performance versus not using BadRank at all in the feature data. This suggests a potential benefit from using BadRank for web spam detection with minimal tuning of parameters.

The differences between the different parameterizations of BadRank are relatively small, making it difficult to draw conclusions about which parameters are best. Based on these limited results and the underlying theory, however, we recommend "self links" as the fix for potential substochastic $\mathbf{W}$. It also seems that $\beta = 0.20$ was the best of the choices considered; this corresponds to taking an average of five steps before jumping back to a random node in $\mathcal{B}$. It is interesting that incorporating trust seems to have decreased the

[3] `http://webspam.lip6.fr/`

AUC in these experiments—we have no explanation for this behavior. Also, using $\gamma = 0$ gave a slight boost as compared to $\gamma = 0.01$. It may only be worthwhile to use $\gamma > 0$ if convergence is a problem for the method.

This page intentionally left blank.

4 Related Work

As mentioned previously, Anti-Trust Rank [15] is closely related to BadRank. It uses exactly the formula given in (3) with $\alpha = 0.85$. There is also a similar method called R-SpamRank [17]. Both of these methods were tested and able to show that pages with high BadRank scores were indeed highly likely to be web spam. In those evaluations, however, BadRank was not combined with other features. In our experiments and as mentioned in [9], even though pages with high BadRank scores are likely to be spam, relying on that metric alone is insufficient. This was our motivation for combining BadRank with existing feature data.

The precomputed feature data provided with WEBSPAM-UK2007 is based on features described in [3, 5]. These are a collection of both link- and content- based features. Many other researchers have looked at classification algorithms for link spam using a variety of features; see, e.g., [9, 2, 4, 1, 26]. The general consensus is that a combination of link- and content-based features is ideal.

An analogue of BadRank is TrustRank [11], which is defined as

$$\mathbf{s}_k(i) = \alpha \sum_{j \to i} \frac{\mathbf{s}_{k-1}(j)}{\mathbf{outlinks}(j)} + (1-\alpha)\mathbf{t}(i) \quad \text{for} \quad 1 \le i \le N,$$

where $\mathbf{t}$ is a vector of trusted site (analogous to $\mathbf{b}$ for "bad" sites). Many papers have considered combining trust and distrust using various combinations of TrustRank and BadRank. In some cases, different methods for *splitting* the scores are considered. Rather than dividing by the number of inlinks (for BadRank) or outlinks (for TrustRank), several papers [24, 18, 19] consider other splitting methods and various combinations of trust and distrust for scoring web sites. The resulting iteration matrices are no longer stochastic, and these methods might be better described as link propagation methods. We note, however, that the idea of biasing the random walk so that a random surfer is more likely to pick a trustworthy page than a less-trustworthy one is mentioned in [18]; this is akin to our idea of graduated trust.

There are many general methods for propagating trust and distrust; see, e.g., [10, 27, 28]. In these cases, there is a more general notion of separate links that convey either trust or distrust. The PageTrust algorithm [7] incorporates both trust and distrust into a single matrix and then proceeds to compute a PageRank-like algorithm, but the iteration matrix is not stochastic and so there is no guarantee of convergence.

This page intentionally left blank.

5 Conclusions

In this paper we described the conditions under which BadRank is guaranteed to converge, including several alternatives for modifying the hyperlink matrix to eliminate leaf nodes which would cause the resulting iteration matrix to be substochastic.

We considered the novel idea of biasing the BadRank scores away from known trusted sites. The simpler idea is to simply remove outbound links from trusted sites to prevent badness from propagating through these nodes. We also consider graduated trust which produces non-uniform probabilities in the random walk that favors nodes that are less trusted.

Computationally, we showed that BadRank can statistically significantly improve the performance of web spam classification. In the best case, the inclusion of BadRank scores as a feature yielded an average 1.7% increase in the classification performance, which is remarkable given that we are adding only one feature to an existing set of 277 features that have been provided by experts in the field. Therefore, we conclude that BadRank is a useful feature to employ in web spam classification.

References

[1] L. Becchetti, C. Castillo, D. Donato, R. Baeza-Yates, and S. Leonardi. Link analysis for web spam detection. *ACM Trans. Web*, 2(1):1–42, 2008.

[2] L. Becchetti, C. Castillo, D. Donato, S. Leonardi, and R. Baeza-Yates. Link-based characterization and detection of web spam. In *AIRWeb'06: Proceedings of the 2nd International Workshop on Adversarial Information Retrieval on the Web*, 2006.

[3] L. Becchetti, C. Castillo, D. Donato, S. Leonardi, and R. Baeza-Yates. Using rank propagation and probabilistic counting for link-based spam detection. In *WebKDD: Proceedings of the Workshop on Web Mining and Web Usage Analysis*. ACM Press, 2006.

[4] A. A. Benczúr, K. Csalogány, and T. Sarlós. Link-based similarity search to fight web spam. In B. D. Davidson, M. Najork, and T. Converse, editors, *AIRWeb'06: Proceedings of the 2nd International Workshop on Adversarial Information Retrieval on the Web*, pages 9–16, 2006.

[5] C. Castillo, D. Donato, A. Gionis, V. Murdock, and F. Silvestri. Know your neighbors: web spam detection using the web topology. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 423–430. ACM, 2007.

[6] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[7] C. de Kerchove and P. Van Dooren. The PageTrust algorithm: How to rank web pages when negative links are allowed? In *SDM08: Proceedings of the 2008 SIAM International Conference on Data Mining*, pages 346–352, 2008.

[8] T. G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

[9] I. Drost and T. Scheffer. Thwarting the nigritude ultramarine: Learning to identify link spam. In *ECML 2005: Proceedings of the 16th European Conference on Machine Learning*, volume 3720 of *Lecture Notes in Computer Science*, pages 96–107. Springer, 2005.

[10] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *WWW 2004: Proceedings of the 13th international conference on World Wide Web*, pages 403–412, New York, NY, USA, 2004. ACM Press.

[11] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with trustrank. In *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, pages 576–587. VLDB Endowment, 2004.

[12] T. Haveliwala. Topic-sensitive pagerank: a context-sensitive ranking algorithm for web search. *Knowledge and Data Engineering, IEEE Transactions on*, 15(4):784–796, July-Aug. 2003.

[13] M. R. Henzinger, R. Motwani, and C. Silverstein. Challenges in web search engines. *SIGIR Forum*, 36(2):11–22, 2002.

[14] I. C. F. Ipsen and T. M. Selee. Pagerank computation, with special attention to dangling nodes. *SIAM Journal on Matrix Analysis and Applications*, 29(4):1281–1296, 2007.

[15] V. Krishnan and R. Raj. Web spam detection with anti-trust rank. In *AIRWeb 2006: Proceedings of the Second International Workshop on Adversarial Information Retrieval on the Web, Technical Report LU-CSE-06-027, Department of Computer Science and Engineering, Lehigh University, Bethlehem, PA, 18015 USA*, pages 37–40, 2006.

[16] A. N. Langville and C. D. Meyer. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, 2006.

[17] C. Liang, L. Ru, and X. Zhu. R-spamrank: A spam detection algorithm based on link analysis. Available at `http://www.mts.jhu.edu/~marchette/ID08/spamrank.pdf`., 2006.

[18] L. Nie, B. Wu, and B. D. Davison. Incorporating trust into web search. Technical Report LU-CSE-06-034, Department of Computer Science and Engineering, Lehigh University, Bethlehem, PA, 18015, Dec. 2006.

[19] L. Nie, B. Wu, and B. D. Davison. Winnowing wheat from the chaff: propagating trust to sift spam from the web. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 869–870. ACM, 2007.

[20] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: bringing order to the Web. Technical Report 1999-66, Stanford Digital Library Technologies Project, 1999.

[21] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

[22] M. Sobek. PR0 — Google's PageRank 0 penalty. `http://pr.efactory.de/e-pr0.shtml`. Last accessed April 16, 2009.

[23] Y.-M. Wang, M. Ma, Y. Niu, and H. Chen. Spam double-funnel: connecting web spammers with advertisers. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 291–300. ACM, 2007.

[24] B. Wu, V. Goel, and B. Davison. Propagating trust and distrust to demote web spam. In *MTW'06: Proceeding of Models of Trust for the Web Workshop, International World Wide Web Conference*, 2006.

[25] Yahoo! Research. Web spam collections. `http://barcelona.research.yahoo.net/webspam/datasets/`. Crawled by the Laboratory of Web Algorithmics, University of Milan, http://law.dsi.unimi.it/. URLs retrieved 05 2009.

[26] Y.-J. Yang, S.-H. Yang, and B.-G. Hu. Fighting webspam: Detecting spam on the graph via content and link features. In *PAKDD 2008: Proceedings of the 12th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, volume 5012 of *Lecture Notes in Computer Science*, pages 1049–1055. Springer, 2008.

[27] C.-N. Ziegler and G. Lausen. Spreading activation models for trust propagation. In *EEE '04: 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service*, pages 28–31, 2004.

[28] C.-N. Ziegler and G. Lausen. Propagation models for trust and distrust in social networks. *Information Systems Frontiers*, 7(4–5):337–358, Dec. 2005.