Combating Web Spam with TrustRank

Zoltán Gyöngyi
Stanford University
Computer Science Department
Stanford, CA 94305
zoltan@cs.stanford.edu

Hector Garcia-Molina
Stanford University
Computer Science Department
Stanford, CA 94305
hector@cs.stanford.edu

Jan Pedersen
Yahoo! Inc.
701 First Avenue
Sunnyvale, CA 94089
jpederse@yahoo-inc.com

Abstract

Web spam pages use various techniques to achieve higher-than-deserved rankings in a search engine's results. While human experts can identify spam, it is too expensive to manually evaluate a large number of pages. Instead, we propose techniques to semi-automatically separate reputable, good pages from spam. We first select a small set of seed pages to be evaluated by an expert. Once we manually identify the reputable seed pages, we use the link structure of the web to discover other pages that are likely to be good. In this paper we discuss possible ways to implement the seed selection and the discovery of good pages. We present results of experiments run on the World Wide Web indexed by AltaVista and evaluate the performance of our techniques. Our results show that we can effectively filter out spam from a significant fraction of the web, based on a good seed set of less than 200 sites.

1 Introduction

The term *web spam* refers to hyperlinked pages on the World Wide Web that are created with the intention of misleading search engines. For example, a pornography site may spam the web by adding thousands of keywords to its home page, often making the text invisible to humans through ingenious use of color schemes. A search engine will then index the extra keywords, and return the pornography page as an answer to queries that contain some of the keywords. As the added keywords are typically not of strictly adult nature, people searching for other topics will be led to the page. Another web spamming technique is the creation of a large number of bogus web pages, all pointing to a single target page. Since many search engines take into account the number of incoming links in ranking pages, the rank of the target page is likely to increase, and appear earlier in query result sets.

Just as with email spam, determining if a page or group of pages is spam is subjective. For instance, consider a cluster of web sites that link to each other's pages repeatedly. These links may represent useful relationships between the sites, or they may have been created with the express intention of boosting the rank of each other's pages. In general, it is hard to distinguish between these two scenarios.

However, just as with email spam, most people can easily identify the blatant and brazen instances of web spam. For example, most would agree that if much of the text on a page is made invisible to humans (as noted above), and is irrelevant to the main topic of the page, then it was added with the intention to mislead. Similarly, if one finds a page with thousands of URLs referring to hosts like

```
buy-canon-rebel-300d-lens-case.camerasx.com,
buy-nikon-d100-d70-lens-case.camerasx.com,
...,
```

and notices that all host names map to the same IP address, then one would conclude that the page was created to mislead search engines. (The motivation behind URL spamming is that many search engines pay special attention to words in host names and give these words a higher weight than if they had occurred in plain text.)

While most humans would agree on the blatant web spam cases, this does not mean that it is easy for a computer to detect such instances. Search engine companies typically employ staff members who specialize in the detection of web spam, constantly scanning the web looking for offenders. When a spam page is identified, a search engine stops crawling it, and its content is no longer indexed. This spam detection process is very expensive and slow, but is critical to the success of search engines: without the removal of the blatant offenders, the quality of search results would degrade significantly.

Our research goal is to assist the human experts who detect web spam. In particular, we want to identify pages

Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004

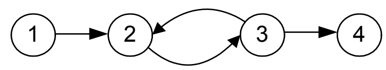

Figure 1: A simple web graph.

and sites that are likely to be spam or that are likely to be reputable. The methods that we present in this paper could be used in two ways: (1) either as helpers in an initial screening process, suggesting pages that should be examined more closely by an expert, or (2) as a counter-bias to be applied when results are ranked, in order to discount possible boosts achieved by spam.

Since the algorithmic identification of spam is very difficult, our schemes do not operate entirely without human assistance. As we will see, the main algorithm we propose receives human assistance as follows. The algorithm first selects a small *seed* set of pages whose "spam status" needs to be determined. A human expert then examines the seed pages, and tells the algorithm if they are spam (*bad* pages) or not (*good* pages). Finally, the algorithm identifies other pages that are likely to be good based on their connectivity with the good seed pages.

In summary, the contributions of this paper are:

1. We formalize the problem of web spam and spam detection algorithms.
2. We define metrics for assessing the efficacy of detection algorithms.
3. We present schemes for selecting seed sets of pages to be manually evaluated.
4. We introduce the TrustRank algorithm for determining the likelihood that pages are reputable.
5. We discuss the results of an extensive evaluation, based on 31 million sites crawled by the AltaVista search engine, and a manual examination of over $2,000$ sites. We provide some interesting statistics on the type and frequency of encountered web contents, and we use our data for evaluating the proposed algorithms.

2 Preliminaries

2.1 Web Model

We model the web as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consisting of a set $\mathcal{V}$ of N pages (vertices) and a set $\mathcal{E}$ of directed links (edges) that connect pages. In practice, a web page p may have multiple HTML hyperlinks to some other page q. In this case we collapse these multiple hyperlinks into a single link $(p,q) \in \mathcal{E}$. We also remove self hyperlinks. Figure 1 presents a very simple web graph of four pages and four links. (For our experiments in Section 6, we will deal with web sites, as opposed to individual web pages. However, our model and algorithms carry through to the case where graph vertices are entire sites.)

Each page has some incoming links, or *inlinks*, and some outgoing links, or *outlinks*. The number of inlinks of a page p is its *indegree* $\iota(p)$, whereas the number of outlinks is its *outdegree* $\omega(p)$. For instance, the indegree of page 3 in Figure 1 is one, while its outdegree is two.

Pages that have no inlinks are called *unreferenced pages*. Pages without outlinks are referred to as *non-referencing pages*. Pages that are both unreferenced and non-referencing at the same time are *isolated pages*. Page 1 in Figure 1 is an unreferenced page, while page 4 is non-referencing.

We introduce two matrix representations of a web graph, which will have important roles in the following sections. One of them is the *transition matrix* $\mathbf{T}$:

$$\mathbf{T}(p,q) = \begin{cases} 0 & \text{if } (q,p) \notin \mathcal{E}, \\ 1/\omega(q) & \text{if } (q,p) \in \mathcal{E}. \end{cases}$$

The transition matrix corresponding to the graph in Figure 1 is:

$$\mathbf{T} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & \frac{1}{2} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \end{pmatrix}.$$

We also define the *inverse transition matrix* $\mathbf{U}$:

$$\mathbf{U}(p,q) = \begin{cases} 0 & \text{if } (p,q) \notin \mathcal{E}, \\ 1/\iota(q) & \text{if } (p,q) \in \mathcal{E}. \end{cases}$$

Note that $\mathbf{U} \neq \mathbf{T}^T$. For the example in Figure 1 the inverse transition matrix is:

$$\mathbf{U} = \begin{pmatrix} 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & \frac{1}{2} & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

2.2 PageRank

PageRank is a well known algorithm that uses link information to assign global importance scores to all pages on the web. Because our proposed algorithms rely on PageRank, this section offers a short overview.

The intuition behind PageRank is that a web page is important if several other important web pages point to it. Correspondingly, PageRank is based on a mutual reinforcement between pages: the importance of a certain page *influences* and is *being influenced* by the importance of some other pages.

The PageRank score $\mathbf{r}(p)$ of a page p is defined as:

$$\mathbf{r}(p) = \alpha \cdot \sum_{q:(q,p)\in\mathcal{E}} \frac{\mathbf{r}(q)}{\omega(q)} + (1-\alpha) \cdot \frac{1}{N},$$

where α is a decay factor.[1] The equivalent matrix equation

[1]Note that there are a number of equivalent definitions of PageRank [12] that might slightly differ in mathematical formulation and numerical properties, but yield the same relative ordering between any two web pages.

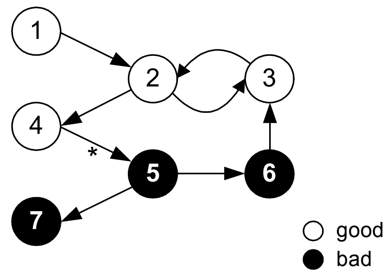

Figure 2: A web of good (white) and bad (black) nodes.

form is:

$$\mathbf{r} = \alpha \cdot \mathbf{T} \cdot \mathbf{r} + (1-\alpha) \cdot \frac{1}{N} \cdot \mathbf{1}_N.$$

Hence, the score of some page p is a sum of two components: one part of the score comes from pages that point to p, and the other (*static*) part of the score is equal for all web pages.

PageRank scores can be computed iteratively, for instance, by applying the Jacobi method [3]. While in a strict mathematical sense, iterations should be run to convergence, it is more common to use only a fixed number of M iterations in practice.

It is important to note that while the regular PageRank algorithm assigns the same static score to each page, a *biased PageRank* version may break this rule. In the matrix equation

$$\mathbf{r} = \alpha \cdot \mathbf{T} \cdot \mathbf{r} + (1-\alpha) \cdot \mathbf{d},$$

vector $\mathbf{d}$ is a *static score distribution vector* of arbitrary, non-negative entries summing up to one. Vector $\mathbf{d}$ can be used to assign a non-zero static score to a set of special pages only; the score of such special pages is then spread during the iterations to the pages they point to.

3 Assessing Trust

3.1 Oracle and Trust Functions

As discussed in Section 1, determining if a page is spam is subjective and requires human evaluation. We formalize the notion of a human checking a page for spam by a binary *oracle function* O over all pages $p \in \mathcal{V}$:

$$\mathsf{O}(p) = \begin{cases} 0 & \text{if } p \text{ is bad,} \\ 1 & \text{if } p \text{ is good.} \end{cases}$$

Figure 2 represents a small seven-page web where good pages are shown as white, and bad pages as black. For this example, calling the oracle on pages 1 through 4 would yield the return value of 1.

Oracle invocations are expensive and time consuming. Thus, we obviously do not want to call the oracle function for all pages. Instead, our objective is to be selective, i.e., to ask a human expert to evaluate only some of the web pages.

To discover good pages without invoking the oracle function on the entire web, we will rely on an important empirical observation we call the *approximate isolation* of the good set: good pages seldom point to bad ones. This notion is fairly intuitive—bad pages are built to mislead search engines, not to provide useful information. Therefore, people creating good pages have little reason to point to bad pages.

However, the creators of good pages can sometimes be “tricked,” so we do find some good-to-bad links on the web. (In Figure 2 we show one such good-to-bad link, from page 4 to page 5, marked with an asterisk.) Consider the following example. Given a good, but unmoderated message board, spammers may include URLs to their spam pages as part of the seemingly innocent messages they post. Consequently, good pages of the message board would link to bad pages. Also, sometimes spam sites offer what is called a *honey pot*: a set of pages that provide some useful resource (e.g., copies of some Unix documentation pages), but that also have hidden links to their spam pages. The honey pot then attracts people to point to it, boosting the ranking of the spam pages.

Note that the converse to approximate isolation does not necessarily hold: spam pages can, and in fact often do, link to good pages. For instance, creators of spam pages point to important good pages either to create a honey pot, or hoping that many good outlinks would boost their hub-score-based ranking [10].

To evaluate pages without relying on O, we will estimate the likelihood that a given page p is good. More formally, we define a *trust function* T that yields a range of values between 0 (bad) and 1 (good). Ideally, for any page p, $\mathsf{T}(p)$ should give us the probability that p is good:

Ideal Trust Property

$$\mathsf{T}(p) = \Pr[\mathsf{O}(p) = 1].$$

To illustrate, let us consider a set of 100 pages and say that the trust score of each of these pages happens to be 0.7. Let us suppose that we also evaluate all the 100 pages with the oracle function. Then, if T works properly, for 70 of the pages the oracle score should be 1, and for the remaining 30 pages the oracle score should be 0.

In practice, it is very hard to come up with a function T with the previous property. However, even if T does not accurately measure the likelihood that a page is good, it would still be useful if the function could at least help us order pages by their likelihood of being good. That is, if we are given a pair of pages p and q, and p has a lower trust score than q, then this should indicate that p is less likely to be good than q. Such a function would at least be useful in ordering search results, giving preference to pages more likely to be good. More formally, then, a desirable property for the trust function is:

Ordered Trust Property

$$\mathsf{T}(p) < \mathsf{T}(q) \Leftrightarrow \Pr[\mathsf{O}(p) = 1] < \Pr[\mathsf{O}(q) = 1],$$
$$\mathsf{T}(p) = \mathsf{T}(q) \Leftrightarrow \Pr[\mathsf{O}(p) = 1] = \Pr[\mathsf{O}(q) = 1].$$

Another way to relax the requirements for T is to introduce a threshold value δ:

Threshold Trust Property

$$\mathsf{T}(p) > \delta \Leftrightarrow \mathsf{O}(p) = 1.$$

That is, if a page p receives a score above δ, we know that it is good. Otherwise, we cannot tell anything about p. Such a function T would at least be capable of telling us that some subset of pages with a trust score above δ is good. Note that a function T with the threshold property does not necessarily provide an ordering of pages based on their likelihood of being good.

3.2 Evaluation Metrics

This section introduces three metrics that help us evaluate whether a particular function T has some of the desired properties.

We assume that we have a sample set $\mathcal{X}$ of web pages for which we can invoke both T and O. Then, we can evaluate how well a desired property is achieved for this set. In Section 6 we discuss how a meaningful sample set $\mathcal{X}$ can be selected, but for now, we can simply assume that $\mathcal{X}$ is a set of random web pages.

Our first metric, *pairwise orderedness*, is related to the ordered trust property. We introduce a binary function $\mathsf{I}(\mathsf{T}, \mathsf{O}, p, q)$ to signal if a bad page received an equal or higher trust score than a good page (a violation of the ordered trust property):

$$\mathsf{I}(\mathsf{T}, \mathsf{O}, p, q) = \begin{cases} 1 & \text{if } \mathsf{T}(p) \geq \mathsf{T}(q) \text{ and } \mathsf{O}(p) < \mathsf{O}(q), \\ 1 & \text{if } \mathsf{T}(p) \leq \mathsf{T}(q) \text{ and } \mathsf{O}(p) > \mathsf{O}(q), \\ 0 & \text{otherwise.} \end{cases}$$

Next, we generate from our sample $\mathcal{X}$ a set $\mathcal{P}$ of ordered pairs of pages (p, q), $p \neq q$, and we compute the fraction of the pairs for which T did not make a mistake:

Pairwise Orderedness

$$\mathsf{pairord}(\mathsf{T}, \mathsf{O}, \mathcal{P}) = \frac{|\mathcal{P}| - \sum_{(p,q) \in \mathcal{P}} \mathsf{I}(\mathsf{T}, \mathsf{O}, p, q)}{|\mathcal{P}|}.$$

Hence, if $\mathsf{pairord}$ equals 1, there are no cases when T misrated a pair. Conversely, if $\mathsf{pairord}$ equals zero, then T misrated all the pairs. In Section 6 we discuss how to select a set $\mathcal{P}$ of sample page pairs for evaluation.

Our next two metrics are related to the threshold trust property. It is natural to think of the performance of function T in terms of the commonly used *precision* and *recall* metrics [1] for a certain threshold value δ. We define precision as the fraction of good among all pages in $\mathcal{X}$ that have a trust score above δ:

Precision

$$\mathsf{prec}(\mathsf{T}, \mathsf{O}) = \frac{|\{p \in \mathcal{X} | \mathsf{T}(p) > \delta \text{ and } \mathsf{O}(p) = 1\}|}{|\{q \in \mathcal{X} | \mathsf{T}(q) > \delta\}|}.$$

Similarly, we define recall as the ratio between the number of good pages with a trust score above δ and the total number of good pages in $\mathcal{X}$:

Recall

$$\mathsf{rec}(\mathsf{T}, \mathsf{O}) = \frac{|\{p \in \mathcal{X} | \mathsf{T}(p) > \delta \text{ and } \mathsf{O}(p) = 1\}|}{|\{q \in \mathcal{X} | \mathsf{O}(q) = 1\}|}.$$

4 Computing Trust

Let us begin our quest for a proper trust function by starting with some simple approaches. We will then combine the gathered observations and construct the TrustRank algorithm in Section 4.3.

Given a limited budget L of O-invocations, it is straightforward to select at random a *seed set* $\mathcal{S}$ of L pages and call the oracle on its elements. (In Section 5 we discuss how to select a better seed set.) We denote the subsets of good and bad seed pages by $\mathcal{S}^+$ and $\mathcal{S}^-$, respectively. Since the remaining pages are not checked by the human expert, we assign them a trust score of $1/2$ to signal our lack of information. Therefore, we call this scheme the *ignorant* trust function T_0, defined for any $p \in \mathcal{V}$ as follows:

Ignorant Trust Function

$$\mathsf{T}_0(p) = \begin{cases} \mathsf{O}(p) & \text{if } p \in \mathcal{S}, \\ 1/2 & \text{otherwise.} \end{cases}$$

For example, we can set L to 3 and apply our method to the example in Figure 2. A randomly selected seed set could then be $\mathcal{S} = \{1, 3, 6\}$. Let $\mathbf{o}$ and $\mathbf{t}_0$ denote the vectors of oracle and trust scores for each page, respectively. In this case,

$$\begin{aligned} \mathbf{o} &= [1, \quad 1, \quad 1, \quad 1, \quad 0, \quad 0, \quad 0], \\ \mathbf{t}_0 &= [1, \quad \tfrac{1}{2}, \quad 1, \quad \tfrac{1}{2}, \quad \tfrac{1}{2}, \quad 0, \quad \tfrac{1}{2}]. \end{aligned}$$

To evaluate the performance of the ignorant trust function, let us suppose that our sample $\mathcal{X}$ consists of all 7 pages, and that we consider all possible $7 \cdot 6 = 42$ ordered pairs. Then, the pairwise orderedness score of T_0 is $17/21$. Similarly, for a threshold $\delta = 1/2$, the precision is 1 while the recall is $1/2$.

4.1 Trust Propagation

As a next step in computing trust scores, we take advantage of the approximate isolation of good pages. We still select at random the set $\mathcal{S}$ of L pages that we invoke the oracle on. Then, expecting that good pages point to other good pages only, we assign a score of 1 to all pages that are reachable from a page in $\mathcal{S}^+$ in M or fewer steps. The appropriate trust function T_M is defined as:

M	pairord	prec	rec
1	19/21	1	3/4
2	1	1	1
3	17/21	4/5	1

Table 1: Performance of the M-step trust function T_M for $M \in \{1,2,3\}$.

***M*-Step Trust Function**

$$\mathsf{T}_M(p) = \begin{cases} \mathsf{O}(p) & \text{if } p \in \mathcal{S}, \\ 1 & \text{if } p \notin \mathcal{S} \text{ and } \exists q \in \mathcal{S}^+ : q \rightsquigarrow_M p, \\ 1/2 & \text{otherwise,} \end{cases}$$

where $q \rightsquigarrow_M p$ denotes the existence of a path of a maximum length of M from page q to page p. Such a path must not include bad seed pages.

Using the example in Figure 2 and the seed set $\mathcal{S} = \{1,3,6\}$, we present the trust score assignments for three different values of M:

$$\begin{array}{lcl} M=1: \ \mathbf{t}_1 & = & [1, \ 1, \ 1, \ \tfrac{1}{2}, \ \tfrac{1}{2}, \ 0, \ \tfrac{1}{2}], \\ M=2: \ \mathbf{t}_2 & = & [1, \ 1, \ 1, \ 1, \ \tfrac{1}{2}, \ 0, \ \tfrac{1}{2}], \\ M=3: \ \mathbf{t}_3 & = & [1, \ 1, \ 1, \ 1, \ 1, \ 0, \ \tfrac{1}{2}]. \end{array}$$

We would expect that T_M performs better than T_0 with respect to some of our metrics. Indeed, Table 1 shows that for $M=1$ and $M=2$, both pairwise orderedness and recall increase, and precision remains 1. However, there is a drop in performance when we go to $M=3$. The reason is that page 5 receives a score of 1 due to the link from good page 4 to bad page 5 (marked with an asterisk on Figure 2).

As we saw in the previous example, the problem with M-step trust is that we are not absolutely sure that pages reachable from good seeds are indeed good. As a matter of fact, the further away we are from good seed pages, the less certain we are that a page is good. For instance, in Figure 2 there are 2 pages (namely, pages 2 and 4) that are at most 2 links away from the good seed pages. As both of them are good, the probability that we reach a good page in at most 2 steps is 1. Similarly, the number of pages reachable from the good seed in at most 3 steps is 3. Only two of these (pages 2 and 4) are good, while page 5 is bad. Thus, the probability of finding a good page drops to $2/3$.

4.2 Trust Attenuation

These observations suggest that we reduce trust as we move further and further away from the good seed pages. There are many ways to achieve this attenuation of trust. Here we describe two possible schemes.

Figure 3 illustrates the first idea, which we call *trust dampening*. Since page 2 is one link away from the good seed page 1, we assign it a dampened trust score of β, where $\beta < 1$. Since page 3 is reachable in one step from page 2 with score β, it gets a dampened score of $\beta \cdot \beta$.

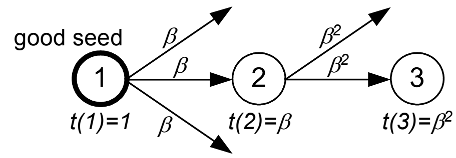

Figure 3: Trust dampening.

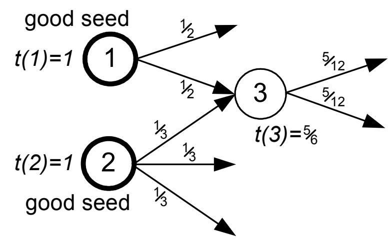

Figure 4: Trust splitting.

We also need to decide how to assign trust to pages with multiple inlinks. For instance, in Figure 3, assume page 1 also links to page 3. We could assign page 3 the maximum trust score, in this case β, or the average score, in this case $(\beta + \beta \cdot \beta)/2$.

The second technique for trust attenuation, which we call *trust splitting*, is based on the following observation: the care with which people add links to their pages is often inversely proportional to the number of links on the page. That is, if a good page has only a handful of outlinks, then it is likely that the pointed pages are also good. However, if a good page has hundreds of outlinks, it is more probable that some of them will point to bad pages.

This observation leads us to splitting trust as it propagates to other pages: if page p has a trust score of $\mathsf{T}(p)$ and it points to $\omega(p)$ pages, each of the $\omega(p)$ pages will receive a score fraction $\mathsf{T}(p)/\omega(p)$ from p. In this case, the actual score of a page will be the sum of the score fractions received through its inlinks. Intuitively, the more "credit" a page accumulates from some other pages, the more probable that it is good. (We can normalize summed scores to our standard range of $[0,1]$.)

Figure 4 illustrates trust splitting. Good seed page 1 has two outlinks, so it distributes half of its score of 1 to both pages it points to. Similarly, good seed page 2 has three outlinks, so each page it points to receives one third of its score. The score of page 3 will then be $1/2 + 1/3 = 5/6$.

Notice that we can also combine trust splitting with dampening. In Figure 4, for instance, page 3 could receive a score of $\beta \cdot (1/2 + 1/3)$.

There are multiple ways of implementing trust dampening and/or splitting. In the next section we present one implementation that shares the same mathematical formulation with a biased PageRank computation in M steps. This feature means that we can rely on PageRank code (with minor changes) to compute trust scores. The resulting advan-

function TrustRank
input
  $\mathbf{T}$ — transition matrix
  N — number of pages
  L — limit of oracle invocations
  α_B — decay factor for biased PageRank
  M_B — number of biased PageRank iterations
output
  $\mathbf{t}^*$ — TrustRank scores
begin
  // evaluate seed-desirability of pages
(1) $\mathbf{s} = \mathsf{SelectSeed}(\ldots)$
  // generate corresponding ordering
(2) $\sigma = \mathsf{Rank}(\{1,\ldots,N\}, \mathbf{s})$
  // select good seeds
(3) $\mathbf{d} = \mathbf{0}_N$
  for $i = 1$ to L do
    if $\mathsf{O}(\sigma(i)) == 1$ then
      $\mathbf{d}(\sigma(i)) = 1$
  // normalize static score distribution vector
(4) $\mathbf{d} = \mathbf{d}/|\mathbf{d}|$
  // compute TrustRank scores
(5) $\mathbf{t}^* = \mathbf{d}$
  for $i = 1$ to M_B do
    $\mathbf{t}^* = \alpha_B \cdot \mathbf{T} \cdot \mathbf{t}^* + (1 - \alpha_B) \cdot \mathbf{d}$
  return $\mathbf{t}^*$
end

Figure 5: The TrustRank algorithm.

tage is important since substantial effort has been spent on making PakeRank computations efficient with very large data sets (for instance, see [5, 8]).

4.3 The TrustRank Algorithm

Function TrustRank, shown in Figure 5, computes trust scores for a web graph. We explain the algorithm by walking through its execution on Figure 2.

The input to the algorithm is the graph (the transition matrix $\mathbf{T}$ and the number N of web pages) and parameters that control execution (L, M_B, α_B, see below).

As a first step, the algorithm calls function SelectSeed, which returns a vector $\mathbf{s}$. The entry $\mathbf{s}(p)$ in this vector gives the "desirability" of page p as a seed page. (Please refer to Section 5 for details.) As we will see in Section 5.1, one version of SelectSeed returns the following vector on the example of Figure 2:

$$\mathbf{s} = \begin{bmatrix}0.08, & 0.13, & 0.08, & 0.10, & 0.09, & 0.06, & 0.02\end{bmatrix}.$$

In step (2) function $\mathsf{Rank}(\mathbf{x}, \mathbf{s})$ generates a permutation $\mathbf{x}'$ of the vector $\mathbf{x}$, with elements $\mathbf{x}'(i)$ in decreasing order of $\mathbf{s}(\mathbf{x}'(i))$. In other words, Rank reorders the elements of $\mathbf{x}$ in decreasing order of their $\mathbf{s}$-scores. For our example, we get:

$$\sigma = \begin{bmatrix}2, & 4, & 5, & 1, & 3, & 6, & 7\end{bmatrix}.$$

That is, page 2 is the most desirable seed page, followed by page 4, and so on.

Step (3) invokes the oracle function on the L most desirable seed pages. The entries of the static score distribution vector $\mathbf{d}$ that correspond to good seed pages are set to 1.

Step (4) normalizes vector $\mathbf{d}$ so that its entries sum up to 1. Assuming that $L = 3$, the seed set is $\{2, 4, 5\}$. Pages 2 and 4 are the good seeds, and we get the following static score distribution vector for our example:

$$\mathbf{d} = \begin{bmatrix}0, & \frac{1}{2}, & 0, & \frac{1}{2}, & 0, & 0, & 0\end{bmatrix},$$

Finally, step (5) evaluates TrustRank scores using a biased PageRank computation with $\mathbf{d}$ replacing the uniform distribution. Note that step (5) implements a particular version of trust dampening and splitting: in each iteration, the trust score of a node is split among its neighbors and dampened by a factor α_B.

Assuming that $\alpha_B = 0.85$ and $M_B = 20$, the algorithm computes the following result:

$$\mathbf{t}^* = \begin{bmatrix}0, & 0.18, & 0.12, & 0.15, & 0.13, & 0.05, & 0.05\end{bmatrix}.$$

Notice that because of the way we iteratively propagate trust scores, the good seed pages (namely, 2 and 4) no longer have a score of 1. However, they still have the highest scores. Also notice that good seed page 4 has a lower score than good seed page 2. This is due to the link structure in this example: page 2 has an inlink from a high scoring page (page 3), while page 4 does not. Thus, our TrustRank algorithm "refines" the original scores given by the oracle, determining that there is even more evidence that page 2 is good as compared to 4. If desired, one can normalize the resulting vector by dividing all scores by the highest score (making the score of page 2 equal to one), but this operation does not change the relative ordering of the pages.

We see in this example that the TrustRank algorithm usually gives good pages a higher score. In particular, three of the four good pages (namely, pages 2, 3, and 4) got high scores and two of the three bad pages (pages 6 and 7) got low scores. However, the algorithm failed to assign pages 1 and 5 adequate scores. Page 1 was not among the seeds, and it did not have any inlinks through which to accumulate score, so its score remained at 0. All good unreferenced web pages receive a similar treatment, unless they are selected as seeds. Bad page 5 received a high score because it is the direct target of one of the rare good-to-bad links. As we will see in Section 6, in spite of errors like these, on a real web graph the TrustRank algorithm is still able to correctly identify a significant number of good pages.

5 Selecting Seeds

The goal of function SelectSeed is to identify desirable pages for the seed set. That is, we would like to find pages

```
function SelectSeed
input
    U     inverse transition matrix
    N     number of pages
    α_I   decay factor
    M_I   number of iterations
output
    s     inverse PageRank scores
begin
    s = 1_N
    for i = 1 to M do
        s = α · U · s + (1 − α) · (1/N) · 1_N
    return s
end
```

Figure 6: The inverse PageRank algorithm.

that will be the most useful in identifying additional good pages. At the same time, we want to keep the seed set reasonably small to limit the number of oracle invocations. In this section we discuss two strategies for SelectSeed, in addition to the random selection strategy that was mentioned earlier.

5.1 Inverse PageRank

Since trust flows out of the good seed pages, one approach is to give preference to pages from which we can reach many other pages. In particular, we could select seed pages based on the number of outlinks. For instance, considering our example in Figure 2, the appropriate seed set of $L = 2$ pages would be $\mathcal{S} = \{2, 5\}$, since pages 2 and 5 have the largest number of outlinks (namely two).

Following a similar reasoning, the coverage can be improved even further. We can build the seed set from those pages that point to many pages that in turn point to many pages and so on. Interestingly, this approach leads us to a scheme closely related PageRank—the difference is that in our case the importance of a page depends on its outlinks, not its inlinks. Therefore, to compute the desirability of a page, we perform a PageRank computation on the graph $\mathcal{G}' = (\mathcal{V}, \mathcal{E}')$, where

$$(p, q) \in \mathcal{E}' \Leftrightarrow (q, p) \in \mathcal{E}.$$

Since we inverted the links, we call our algorithm *inverse PageRank*.

Figure 6 shows a SelectSeed algorithm that performs the inverse PageRank computation. Note that the decay factor α_I and the number of iterations M_I can be different from the values α_B and M_B used by the TrustRank algorithm. The computation is identical to that in the traditional PageRank algorithm (Section 2.2), except that the inverse transition matrix $\mathbf{U}$ is used instead of the regular transition matrix $\mathbf{T}$.

For our example from Figure 2, the inverse PageRank algorithm ($\alpha_I = 0.85, M_I = 20$) yields the following scores

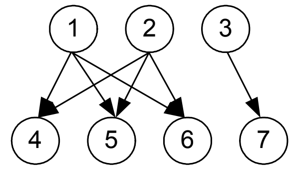

Figure 7: A graph for which inverse PageRank does not yield maximum coverage.

(already shown in Section 4.3):

$$\mathbf{s} = \begin{bmatrix} 0.08, & 0.13, & 0.08, & 0.10, & 0.09, & 0.06, & 0.02 \end{bmatrix}.$$

For a value of $L = 3$, the seed set is $\mathcal{S} = \{2, 4, 5\}$. Correspondingly, the good seed set is $\mathcal{S}^+ = \{2, 4\}$, so pages 2 and 4 are used as starting points for score distribution.

It is important to note that inverse PageRank is a heuristic (that works well in practice, as we will see in Section 6). First, inverse PageRank does not guarantee maximum coverage. For instance, in the example in Figure 7 and for $L = 2$, maximum coverage is achieved through the seed set $\{1, 3\}$ or $\{2, 3\}$. However, the inverse PageRank computation yields the score vector:

$$\mathbf{s} = \begin{bmatrix} 0.05, & 0.05, & 0.04, & 0.02, & 0.02, & 0.02, & 0.02 \end{bmatrix},$$

which leads to the seed set $\mathcal{S} = \{1, 2\}$.

Nevertheless, inverse PageRank is appealing because its execution time is polynomial in the number of pages, while determining the maximum coverage is an $\mathcal{NP}$-complete problem.[2]

A second reason why inverse PageRank is a heuristic is that maximizing coverage may not always be the best strategy. To illustrate, let us propagate trust via splitting, without any dampening. Returning to Figure 7, say we only select page 2 as seed and it turns out to be good. Then pages 4, 5, and 6 each receive a score of $1/3$. Now, assume we only select page 3 as seed and it also happens to be good. Then page 7 gets a score of 1. Depending on our ultimate goal, it may be preferable to use page 3, since we can be more certain about the page it identifies, even if the set is smaller. However, if we are only using trusts scores for comparing against other trust scores, it may still be better to learn about more pages, even if with less absolute accuracy.

5.2 High PageRank

So far we have assumed that the value of identifying a page as good or bad is the same for all web pages. Yet, it may be

[2]The general problem of identifying the minimal set of pages that yields maximum coverage is equivalent to the independent set problem [7] on directed graphs as shown next. The web graph can be transformed in a directed graph $\mathcal{G}'' = (\mathcal{V}, \mathcal{E}'')$, where an edge $(p, q) \in \mathcal{E}''$ signals that page q can be reached from page p. We argue that such transformation does not change the complexity class of the algorithm, since it involves breadth-first search that has polynomial execution time. Then, finding a minimal set that provides maximum coverage is the same as finding the maximum independent set for $\mathcal{G}''$, which is an $\mathcal{NP}$-complete problem.

more important to ascertain the goodness of pages that will appear high in query result sets. For example, say we have four pages p, q, r, and s, whose contents match a given set of query terms equally well. If the search engine uses PageRank to order the results, the page with highest rank, say p, will be displayed first, followed by the page with next highest rank, say q, and so on. Since it is more likely the user will be interested in pages p and q, as opposed to pages r and s (pages r and s may even appear on later result pages and may not even be seen by the user), it seems more useful to obtain accurate trust scores for pages p and q rather than for r and s. For instance, if page p turns out to be spam, the user may rather visit page q instead.

Thus, a second heuristic for selecting a seed set is to give preference to pages with high PageRank. Since high-PageRank pages are likely to point to other high-PageRank pages, then good trust scores will also be propagated to pages that are likely to be at the top of result sets. Thus, with PageRank selection of seeds, we may identify the goodness of fewer pages (as compared to inverse PageRank), but they may be more important pages to know about.

6 Experiments

6.1 Data Set

To evaluate our algorithms, we performed experiments using the complete set of pages crawled and indexed by the AltaVista search engine as of August 2003.

In order to reduce computational demands, we decided to work at the level of web sites instead of individual pages. (Note that all presented methods work equally well for either pages or sites.) We grouped the several billion pages into $31,003,946$ sites, using a proprietary algorithm that is part of the AltaVista engine. Although the algorithm relies on several heuristics to fine-tune its decisions, roughly speaking, all individual pages that share a common fully qualified host name[3] become part of the same site. Once we decided on the sites, we added a single link from site a to site b if in the original web graph there were one or more links from pages of site a pointing to pages of site b.

One interesting fact that we have noticed from the very beginning was that more than one third of the sites ($13,197,046$) were unreferenced. Trust propagation algorithms rely on inlink information, so are unable to differentiate among these sites without inlinks. Fortunately, the unreferenced sites are ranked low in query results (receive an identical, minimal static PageRank score), so it is not critical to separate good and bad sites among them.

For our evaluations, the first author of this paper played the role of the oracle, examining pages of various sites, determining if they are spam, and performing additional classification, as we will see. Of course, using an author as an evaluator raises the issue of bias in the results. However, this was our only choice. Our manual evaluations took weeks: checking a site involves looking at many of its pages and also the linked sites to determine if there is an intention to deceive search engines. Finding an expert working at one of the very competitive search engine companies who was knowledgeable enough and had time for this work was next to impossible. Instead, the first author spent time looking over the shoulder of the experts, learning how they identified spam sites. Then, he made every effort to be unbiased and to apply the experts' spam detection techniques.

6.2 Seed Set

As a first step, we conducted experiments to compare the inverse PageRank and the high PageRank seed selection schemes described in Sections 5.1 and 5.2, respectively. In order to be able to perform the comparison quickly, we ran our experiments on synthetic web graphs that capture the essential spam-related features of the web. We describe these experiments in [4]. Due to space limitations, here we just note that inverse PageRank turned out to be slightly better at identifying useful seed sets. Thus, for the rest of our experiments on the full, real web, we relied on the inverse PageRank method.

In implementing seed selection using inverse PageRank, we fine-tuned the process in order to streamline the oracle evaluations. First, we performed a full inverse PageRank computation on the site-level web graph, using parameters $\alpha_I = 0.85$ and $M_I = 20$. (The decay factor of 0.85 was first reported in [12] and has been regarded as the standard in PageRank literature ever since. Our tests showed that 20 iterations were enough to achieve convergence on the relative ordering of the sites.)

After ordering the sites based on their inverse PageRank scores (step (2) in Figure 5), we focused our attention on the top $25,000$. Instead of a full oracle evaluation of these sites, we first did a cursory evaluation to eliminate some problematic ones. In particular, we noticed that sites with highest inverse PageRank scores showed a heavy bias toward spam, due to the presence of *Open Directory clones*: some spammers duplicate the entire content of the DMOZ Open Directory either in the hope of increasing their hub score [10] or with the intention of creating honey pots, as discussed in Section 3.1. In order to get rid of the spam quickly, we removed from our list of $25,000$ sites all that were not listed in any of the major web directories, reducing the initial set to roughly $7,900$. By sampling the sites that were filtered out, we found that insignificantly few reputable ones were removed by the process.

Out of the remaining $7,900$ sites, we manually evaluated the top $1,250$ (seed set $\mathcal{S}$) and selected 178 sites to be used as good seeds. This procedure corresponded to step (3) in Figure 5. The relatively small size of the good seed set $\mathcal{S}^+$ is due to the extremely rigorous selection criteria that we adopted: not only did we make sure that the sites were not

[3]The *fully qualified host name* is the portion of the URL between the `http://` prefix, called the *scheme*, and the first slash character that usually follows the top level domain, such as `.com`, or the server's TCP port number. For instance, the fully qualified host name for the URL `http://www-db.stanford.edu/db_pages/members.html` is `www-db.stanford.edu`.

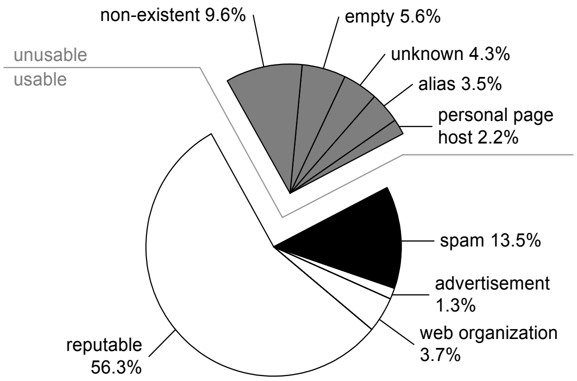

Figure 8: Composition of the evaluation sample.

spam, but we also applied a second filter—we only selected sites with a clearly identifiable authority (such as a governmental or educational institution or company) that controlled the contents of the site. The extra filter was added to guarantee the longevity of the good seed set, since the presence of physical authorities decreases the chance that the sites would degrade in the short run.

6.3 Evaluation Sample

In order to evaluate the metrics presented in Section 3.2, we needed a set $\mathcal{X}$ of sample sites with known oracle scores. (Note that this is different from the seed set and it is only used for assessing the performance of our algorithms.) We settled on a sample of 1000 sites, a number that gave us enough data points, and was still manageable in terms of oracle evaluation time.

We decided *not* to select the 1000 sample sites of $\mathcal{X}$ at random. With a random sample, a great number of the sites would be very small (with few pages) and/or have very low PageRank. (Both size and PageRank follow power-law distributions, with many sites at the tail end of the distribution.) As we discussed in Section 5.2, it is more important for us to correctly detect spam in high PageRank sites, since they will more often appear high in query result sets. Furthermore, it is hard for the oracle to evaluate small sites due to the reduced body of evidence, so it also does not make sense to consider many small sites in our sample.

In order to assure diversity, we adopted the following sampling method. We generated the list of sites in decreasing order of their PageRank scores, and we segmented it into 20 buckets. Each of the buckets contained a different number of sites, with scores summing up to 5 percent of the total PageRank score. Therefore, the first bucket contained the 86 sites with the highest PageRank scores, bucket 2 the next 665, while the 20th bucket contained 5 million sites that were assigned the lowest PageRank scores.

We constructed our sample set of 1000 sites by selecting 50 sites at random from each bucket. Then, we performed a manual (oracle) evaluation of the sample sites, determining if they were spam or not. The outcome of the evaluation process is presented in Figure 8, a pie-chart that shows the way our sample breaks down to various types of sites. We found that we could use 748 of the sample sites to evaluate TrustRank:

- *Reputable.* 563 sites featured quality contents with zero or a statistically insignificant number of links pointing to spam sites.
- *Web organization.* 37 sites belonged to organizations that either have a role in the maintenance of the World Wide Web or perform business related to Internet services. While all of them were good sites, most of their links were automatic (e.g., “Site hosted by Provider X”). Therefore, we decided to give them a distinct label to be able to follow their features separately.
- *Advertisement.* 13 of the sites were ones acting as targets for banner ads. These sites lack real useful content and their high PageRank scores are due exclusively to the large number of automatic links that they receive. Nevertheless, they still qualify as good sites without any sign of spamming activity.
- *Spam.* 135 sites featured various forms of spam. We considered these sites as bad ones.

These 748 sites formed our sample set $\mathcal{X}$. The remaining 252 sites were deemed unusable for the evaluation of TrustRank for various reasons:

- *Personal page host.* 22 of the sites hosted personal web pages. The large, uncontrolled body of editors contributing to the wide variety of contents for each of these sites made it impossible to categorize them as either bad or good. Note that this issue would not appear in a page-level evaluation.
- *Alias.* 35 sites were simple aliases of sites better known under a different name. We decided to drop these aliases because the importance of the alias could not reflect the importance of the original site appropriately.
- *Empty.* 56 sites were empty, consisting of a single page that provided no useful information.
- *Non-existent.* 96 sites were non-existent—either the DNS lookup failed, or our systems were not able to establish a TCP/IP connection with the corresponding computers.
- *Unknown.* We were unable to properly evaluate 43 sites based on the available information. These sites were mainly East Asian ones, which represented a challenge because of the lack of English translation.

6.4 Results

In Section 4 we described a number of strategies for propagating trust from a set of good seeds. In this section we focus on three of the alternatives, TrustRank and two baseline strategies, and evaluate their performance using our sample $\mathcal{X}$:

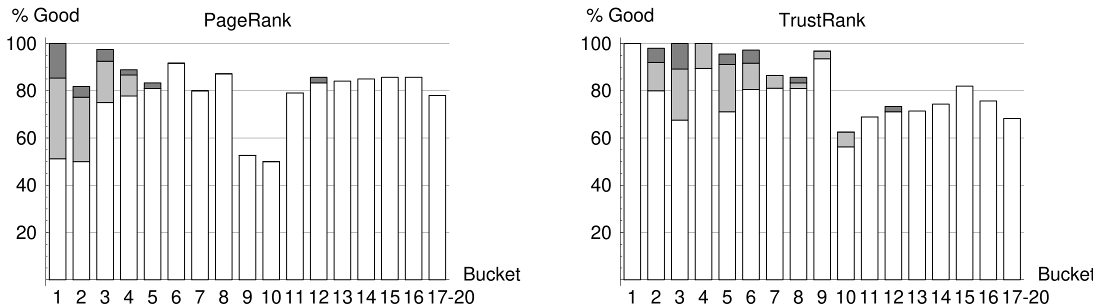

Figure 9: Good sites in PageRank and TrustRank buckets.

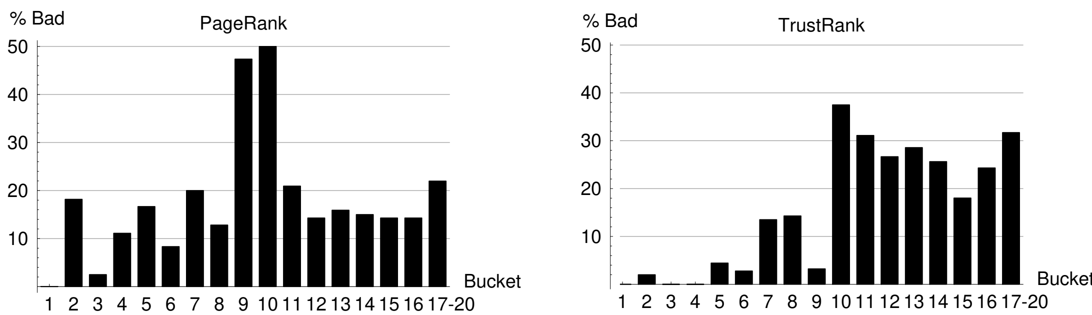

Figure 10: Bad sites in PageRank and TrustRank buckets.

1. *TrustRank.* We used the algorithm in Figure 5 ($M_B = 20$ iterations and decay factor of $\alpha_B = 0.85$) and our selected 178 good seeds.
2. *PageRank.* PageRank was originally considered highly resilient to spamming because it measures global importance (limited, local changes to the link structure have low impact on the scores). Thus, it is natural to ask how well PageRank can cope with spam in today's world. Thus, for this alternative we simply used the PageRank of site a as the value of $\mathsf{T}(a)$. We again performed $M = 20$ iterations, with a decay factor of $\alpha = 0.85$.
3. *Ignorant Trust.* As another baseline, we generated the ignorant trust scores of sites. All sites were assigned an ignorant trust score of $1/2$, except for the 1250 seeds, which received scores of 0 or 1.

6.4.1 PageRank versus TrustRank

Let us discuss the difference between PageRank and TrustRank first. Remember, the PageRank algorithm does not incorporate any knowledge about the quality of a site, nor does it explicitly penalize badness. In fact, we will see that it is not very uncommon that some site created by a skilled spammer receives high PageRank score. In contrast, our TrustRank is meant to differentiate good and bad sites: we expect that spam sites were not assigned high TrustRank scores.

Figures 9 and 10 provide a side-by-side comparison of PageRank and TrustRank with respect to the ratio of good and bad sites in each bucket. PageRank buckets were introduced in Section 6.3; we defined TrustRank buckets as containing the same number of sites as PageRank buckets. Note that we merged buckets 17 through 20 both for PageRank and TrustRank. (These last 4 buckets contained the more than 13 million sites that were unreferenced. All such sites received the same minimal static PageRank score and a zero TrustRank score, making it impossible to set up an ordering among them.)

The horizontal axes of Figures 9 and 10 mark the PageRank and TrustRank bucket numbers, respectively. The vertical axis of the first figure corresponds to the percentage of good within a specific bucket, i.e., the number of good sample sites divided by the total number of sample sites in that bucket. Note that reputable, advertisement, and web organization sites all qualify as good ones; their relative contributions are shown by white, middle gray, and dark gray segments, respectively. The vertical axis of the second figure corresponds to the percentage of bad within a specific bucket. For instance, we can derive from Figure 10 that 31% of the usable sample sites in TrustRank bucket 11 are bad ones.

From these figures we see that TrustRank is a reasonable spam detection tool. In particular, note that there is virtually no spam in the top 5 TrustRank buckets, while there is a marked increase in spam concentration in the lower buckets. At the same time, it is surprising that almost 20% of the second PageRank bucket is bad. For PageRank, the proportion of bad sites peaks in buckets 9 and 10 (50% spam), indicating that probably this is as high as average spammers

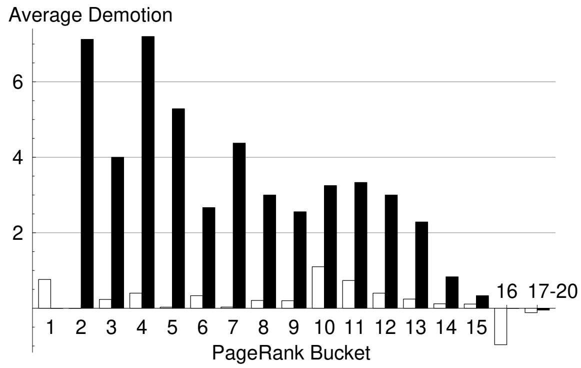

Figure 11: Bucket-level demotion in TrustRank.

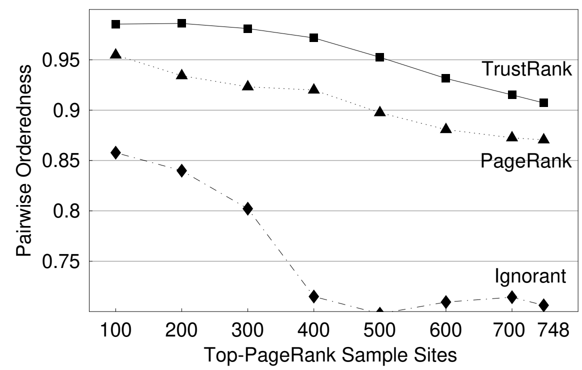

Figure 12: Pairwise orderedness.

could push their sites.

Figure 11 offers another view on the relationship between PageRank and TrustRank. It introduces the notion of *demotion*, the phenomenon that a certain site from a higher PageRank bucket appears in a lower TrustRank bucket. Negative demotion is *promotion*, the case when a site from a lower PageRank bucket shows up in a higher TrustRank bucket. The average demotion of bad sites is an important way to evaluate TrustRank as it shows its success (or lack thereof) to cut the importance of bad sites.

The horizontal axis of Figure 11 stands for PageRank buckets. The vertical axis shows the number of buckets by which sites from a specific PageRank bucket got demoted in TrustRank on average. White bars represent the reputable sites, while black ones denote spam. (Note that we do not show advertisement and web organization sites in the figure.)

As an illustration, we can derive from Figure 11 that spam sites in PageRank bucket 2 got demoted seven buckets on average, thus landing somewhere around TrustRank bucket 9. An example of promotion can be seen in PageRank bucket 16, where good sites appear on average one bucket higher in the TrustRank ordering.

This figure again shows well that TrustRank effectively removes most of the spam from among the top-scored sites. Furthermore, it also reveals that good sites retain their original bucket position in most of the cases. Consequently, we argue that (opposed to PageRank) TrustRank guarantees that top-scored sites are good ones. We also assert that TrustRank is unable to effectively separate low-scored good sites from bad ones, due to the lack of distinguishing features (inlinks) of the sites.

6.4.2 Pairwise Orderedness

We used the pairwise orderedness metric presented in Section 3.2 to evaluate TrustRank with respect to the ordered trust property. For this experiment, we built the set $\mathcal{P}$ of all possible pairs of sites for several subsets of our evaluation sample $\mathcal{X}$. We started by using the subset of $\mathcal{X}$ of the 100 sites with highest PageRank scores, in order to check TrustRank for the most important sites. Then, we gradually added more and more sites to our subset, in their decreasing order of PageRank scores. Finally, we used all pairs of all the 748 sample sites to compute the pairwise orderedness score.

Figure 12 displays the results of this experiment. The horizontal axis shows the number of sample sites used for evaluation, while the vertical axis represents the pairwise orderedness scores for the specific sample sizes. For instance, we can conclude that for the 500 top-PageRank sample sites TrustRank receives a pairwise orderedness score of about 0.95.

Figure 12 also shows the pairwise orderedness scores for the ignorant trust function and PageRank. The overlap between our seed set $\mathcal{S}$ and sample set $\mathcal{X}$ is of 5 good sites, so all but five sample sites received a score of $1/2$ from the ignorant trust function. Hence, the pairwise orderedness scores for the ignorant function represent the case when we have almost no information about the quality of the sites. Similarly, pairwise orderedness scores for PageRank illustrate how much the knowledge of importance can help in distinguishing good and bad. As we can see, TrustRank constantly outperforms both the ignorant function and PageRank.

6.4.3 Precision and Recall

Our last set of experimental results, shown in Figure 13, present the performance of TrustRank with respect to the metrics of precision and recall. We used as threshold values δ the borderline TrustRank scores that separated the 17 TrustRank buckets, discussed in Section 6.4.1. The lowest buckets corresponding to each threshold value are presented on the horizontal axis of the figure; we display the precision and recall scores on the vertical. For instance, if the threshold δ is set so that all and only sample sites in TrustRank buckets 1 through 10 are above it, then precision is 0.86 and recall is 0.55.

TrustRank assigned the highest scores to good sites, and the proportion of bad increases gradually as we move to lower scores. Hence, precision and recall manifest an almost linear decrease and increase, respectively. Note the high (0.82) precision score for the whole sample set: such a value would be very uncommon for traditional informa-

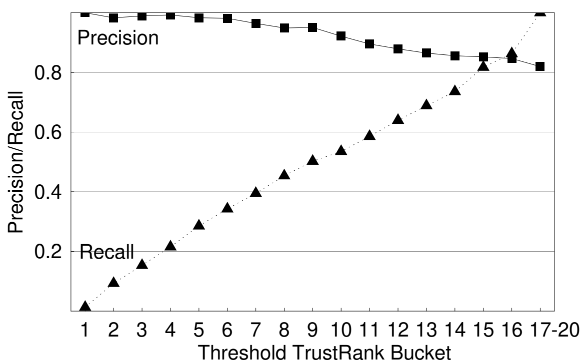

Figure 13: Precision and recall.

tion retrieval problems, where it is usual to have a large corpus of documents, with only a few of those documents being relevant for a specific query. In contrast, our sample set consists most of good documents, all of which are "relevant." This is why the baseline precision score for the sample $\mathcal{X}$ is $613/(613+135) = 0.82$.

7 Related Work

Our work builds on existing PageRank research. The idea of biasing PageRank to combat spam was introduced in [12]. The use of custom static score distribution vectors has been studied in the context of topic-sensitive PageRank [6]. Recent analyses of (biased) PageRank are provided by [2, 11].

The problem of trust has also been addressed in the context of peer-to-peer systems. For instance, [9] presents an algorithm similar to PageRank for computing the reputation or dependability of a node in a peer-to-peer network.

The data mining and machine learning communities also explored the topic of web and email spam detection (for instance, see [13]). However, this research is oriented toward the analysis of individual documents. The analysis typically looks for telltale signs of spamming techniques based on statistics derived from examples.

8 Conclusions

As the web grows in size and value, search engines play an increasingly critical role, allowing users to find information of interest. However, today's search engines are seriously threatened by malicious web spam that attempts to subvert the unbiased searching and ranking services provided by the engines. Search engines are today combating web spam with a variety of ad hoc, often proprietary techniques. We believe that our work is a first attempt at formalizing the problem and at introducing a comprehensive solution to assist in the detection of web spam. Our experimental results show that we can effectively identify a significant number of strongly reputable (non-spam) pages. In a search engine, TrustRank can be used either separately to filter the index, or in combination with PageRank and other metrics to rank search results.

We believe that there are still a number of interesting experiments that need to be carried out. For instance, it would be desirable to further explore the interplay between dampening and splitting for trust propagation. In addition, there are a number of ways to refine our methods. For example, instead of selecting the entire seed set at once, one could think of an iterative process: after the oracle has evaluated some pages, we could reconsider what pages it should evaluate next, based on the previous outcome. Such issues are a challenge for future research.

Acknowledgement

The authors would like to thank David Cossock and Farzin Maghoul for the inspiring discussions and valuable comments.

References

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.

[2] M. Bianchini, M. Gori, and F. Scarselli. Inside PageRank. Tech. rep., University of Siena, 2003.

[3] G. Golub and C. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1996.

[4] Z. Gyöngyi and H. Garcia-Molina. Seed selection in TrustRank. Tech. rep., Stanford University, 2004.

[5] T. Haveliwala. Efficient computation of PageRank. Tech. rep., Stanford University, 1999.

[6] T. Haveliwala. Topic-sensitive PageRank. In *Proceedings of the Eleventh International Conference on World Wide Web*, 2002.

[7] J. Hopcroft, R. Motwani, and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 2001.

[8] S. Kamvar, T. Haveliwala, C. Manning, and G. Golub. Extrapolation methods for accelerating PageRank computations. In *Proceedings of the Twelfth International Conference on World Wide Web*, 2003.

[9] S. Kamvar, M. Schlosser, and H. Garcia-Molina. The EigenTrust algorithm for reputation management in P2P networks. In *Proceedings of the Twelfth International Conference on World Wide Web*, 2003.

[10] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[11] A. Langville and C. Meyer. Deeper inside PageRank. Tech. rep., North Carolina State University, 2003.

[12] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Tech. rep., Stanford University, 1998.

[13] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A Bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, 1998.