Using Rank Propagation and Probabilistic Counting for Link-Based Spam Detection

Luca Becchetti[1] becchett@dis.uniroma1.it
Carlos Castillo[1] castillo@dis.uniroma1.it
Debora Donato[1] donato@dis.uniroma1.it
Stefano Leonardi[1] leon@dis.uniroma1.it
Ricardo Baeza-Yates[2] ricardo@baeza.cl

[1] DIS - Università di Roma "La Sapienza" Rome, Italy

[2] Yahoo! Research Barcelona, Spain & Santiago, Chile

ABSTRACT

This paper describes a technique for automating the detection of Web link spam, that is, groups of pages that are linked together with the sole purpose of obtaining an undeservedly high score in search engines. The problem of Web spam is widespread and difficult to solve, mostly due to the large size of web collections that makes many algorithms unfeasible in practice.

For spam detection we apply only link-based methods, that is, we only study the topology of the Web graph without looking at the contents of the pages. We compute Web page attributes applying rank propagation and probabilistic counting over the Web graph. These attributes are used to build a classifier that is tested over a large collection of Web link spam. After ten-fold cross-validation, our best classifier can detect about 80% of the spam pages with a rate of false positives of 2%. This is competitive with state-of-the-art spam classifiers that use content attributes, and is the first automatic classifier that achieves this precision using only link data.

1. INTRODUCTION

The Web is nowadays both an excellent medium for sharing information, as well as an attractive platform for delivering products and services. This platform is, to some extent, mediated by search engines in order to meet the needs of users seeking for information. Given the vast amount of information available on the Web, it is customary to answer queries with only a small set of results (typically 10 or 20 pages at most). Search engines must then **rank** Web pages, in order to create a short list of high-quality results for users.

The Web contains numerous profit-seeking ventures, so there is an economic incentive from Web site owners to rank high in the result lists of search engines. All the deceptive actions that try to increase the ranking of a page in search engines are generally referred to as **Web spam** or **spamdexing** (a portmanteau of "spam" and "index").

A Web search engine must consider that "any evaluation strategy which counts replicable features of web pages is prone to manipulation" [22]. In practice, such manipulation is widespread, and in many cases, successful. The authors of [9] report that : "among the top 20 URLs in our 100 million page PageRank calculation (. . .) 11 were pornographic, and these high positions appear to have all been achieved using the same form of link manipulation".

In general, we want to explore the neighborhood of a page and see if the link structure around it appears to be artificially generated with the purpose of increasing its rank. We also want to verify if this link structure is the result of a bounded amount of work, restricted to a particular zone of the Web graph, under the control of a single agent. This imposes two algorithmic challenges: the first one is how to compute **simultaneously** statistics about the neighborhood of every page in a huge Web graph, and the second is what to do with this information once it is computed, and how to use it to detect Web spam and demote spam pages.

In this paper we adapt two link-based algorithms, and apply them for detecting malicious link structures on large Web graphs. The main contributions of this paper are:

- We introduce a damping function for rank propagation [1] that produces a metric that helps in separating spam from non-spam.
- We present an improved approximate neighborhood counting algorithm [23] for the application of detecting link spam.
- We describe an automatic classifier that uses only link attributes, without looking at the content of the pages, and achieves a precision that is equivalent to that of the best spam classifiers that use content analysis.

Our algorithms are tested on a large sample of the .uk domain where thousands of domains have been inspected and manually classified as spam or non-spam domains. Concerning the algorithmic contribution of our work, we show a

This is a pre-print of a work in progress.
Last updated: June 6, 2006

version of approximating counting that can be seen as a specific instance of rank propagation. This sheds new light and greatly simplify the method proposed in [23], and suggests that the idea of rank propagation may be adopted as a general framework for computing several relevant statistics in large networks. All our algorithms also work in a streaming model of the Web graph. Every computation is performed by repeating a limited number of times a sequential scan of data stored in secondary memory [17]. We also provide a framework in which the algorithms use an amount of main memory in the order of the number of nodes, while one that requires memory in the order of the number of edges may not be feasible. In conclusion, our methods are scalable to deal with Web graphs of any size.

In [4] we study several combinations of the algorithms presented in this paper with other link-based metrics, showing a detailed breakdown of the different techniques and the classification accuracy achieved in Web spam detection. Here we focus on the description and analysis of the two main algorithms (Truncated PageRank and Probabilistic Counting) and for completeness also include results on their general performance on the spam detection task.

This paper is structured as follows. In section 2, we introduce the framework for the analysis, and present a characterization of the pages we want to detect. In section 3, we present a rank-propagation algorithm that calculates a link-based score that disregards paths whose length is below a certain threshold, so that pages participating in a link farm do not contribute so much to the ranking of the target page. In section 4, we show how to use a variant of the approximate neighborhood counting method (ANF) presented in [23] to estimate the number of "supporters" of a given page at different distances.

2. CHARACTERIZING SPAM PAGES

In [15], **spamming** is defined as "any deliberate action that is meant to trigger an unjustifiably favorable relevance or importance for some web page, considering the page's true value". A **spam page** is a page that is used for spamming or receives a substantial amount of its score from other spam pages. Another definition of spam, given in [24] is "any attempt to deceive a search engine's relevancy algorithm" or simply "anything that would not be done if search engines did not exist".

There are several techniques for spamming the index of a search engine, or **spamdexing**. A spam page may contain an abnormally high number of keywords, or have other text features that make **content-based spam detection** [21, 8] possible. This is not always the case, for instance, in Figure 1 we show a spam page that looks mostly normal and includes only a few out-links.

The page in Figure 1 is part of a **link farm**. A link farm is a densely connected set of pages, created explicitly with the purpose of deceiving a link-based ranking algorithm [27, 2]. In [27], this is called **collusion**, and is defined as the "manipulation of the link structure by a group of users with the intent of improving the rating of one or more users in the group".

This is the kind of pages we are interested in. The targets of our spam-detection algorithms are the pages that receive most of their ranking by participating in link farms. A page that participates in a link farm may have a high in-degree, but little relationship with the rest of the graph. In Figure 2,

Figure 1: Sometimes a spam page can not be easily detected by content-based analysis.

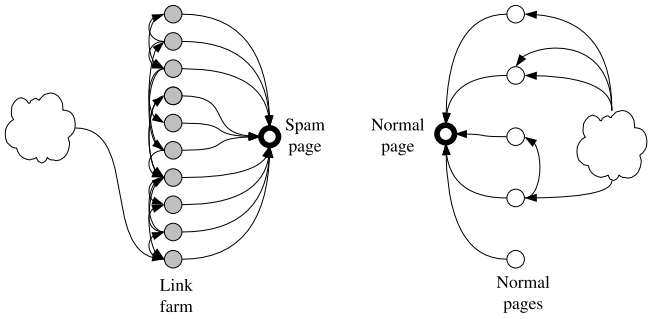


Figure 2: Schematic depiction of the neighborhood of a page participating in a link farm (left) and a normal page (right). A link farm is a densely connected sub-graph, with little relationship with the rest of the Web, but not necessarily disconnected.

we show a schematic diagram depicting the links around a spam page and a normal page. Link farms can receive links by buying advertising, or by buying expired domains used previously by legitimate Web sites.

We must note at this point that there are some types of Web spam that are not completely link-based, and it is very likely that there are some hybrid structures combining link farms (for achieving a high link-based score) and content-based spam, having a few links, to avoid detection. In our opinion, an approach mixing content features, link-based features and user interaction (e.g.: data collected via a toolbar or by observing clicks in search engine results) should work better in practice that a pure link-based method. In this paper, we focus on detecting link farms, but there will be cases in which this type of structure does not exist.

We view our set of Web pages as a **Web graph**, this is, a graph $G = (V, E)$ in which the set V corresponds to Web pages in a subset of the Web, and every link $(x, y) \in E$ corresponds to a hyperlink from page x to page y in the collection. For concreteness, the total number of nodes $N = |V|$ is in the order of 10^{10} [13], and the typical number of links per Web page is between 20 and 30.

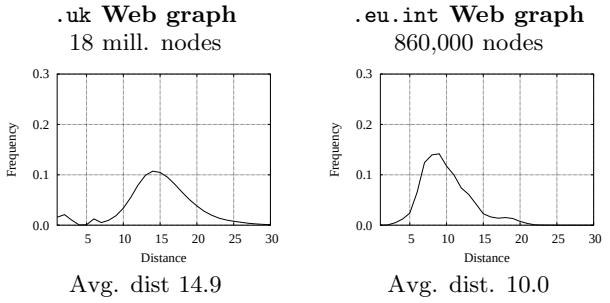


Figure 3: Distribution of the fraction of new supporters found at varying distances (normalized), obtained by backward breadth-first visits from a sample of nodes, in two large Web graphs.

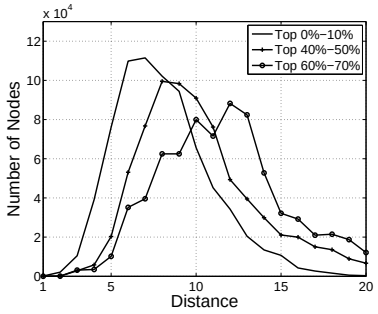


Figure 4: Distribution of the number of new supporters at different distances, for pages in different PageRank buckets.

Link analysis algorithms assume that every link represents an endorsement, in the sense that if there is a link from page x to page y, then the author of page x is recommending page y. Following [5], we call x a **supporter** of page y at distance d, if the shortest path from x to y formed by links in E has length d. The set of supporters of a page are all the other pages that contribute towards its link-based ranking.

As suggested by Figure 2, a particular characteristic of a link farm is that the spam pages might have a large number of distinct supporters at short distances, but this number should be lower than expected at higher distances.

In Figure 3 we plot the distribution of distinct supporters for a random sample nodes in two subsets of the Web obtained from the Laboratory of Web Algorithmics. (All the Web graphs we use in this paper are available from the *Dipartimento di Scienze dell'Informazione*, *Università degli studi di Milano* at `http://law.dsi.unimi.it/`).

We can see that the number of new distinct supporters increases up to a certain distance, between 8 and 12 links in these graphs, and then decreases, as the graph is finite in size and we approach its effective diameter. We expect that the distribution of supporters obtained for a highly-ranked page is different from the distribution obtained for a lowly-ranked page.

To observe this, we calculated the PageRank of the pages in the `eu.int` (European Union) sub-domain. We chose this domain because it is a large, and an entirely spam-free subset of the Web. We grouped the pages into 10 buckets according to their position in the list ordered by PageRank. Figure 4 plots the distribution of supporters for a sample of pages in three of these buckets having high, medium and low ranking respectively.

As expected, highly-ranked pages have a large number of supporters after a few levels, while lowly-ranked pages do not. Note that if two pages belong to the same strongly-connected component of the Web, then eventually their total number of supporters will converge after a certain distance. In that case the areas below the curves are equal.

As shown in Figure 2, we expect that pages participating in a link-farm present anomalies in their distribution of supporters. A major issue is that computing this distribution for all the nodes in a large Web graph is computationally very expensive. A straightforward, computationally unfeasible, approach is to repeat a reverse breadth-first search from each node of the graph, and marking nodes as they are visited [18]. A possible solution could be to compute the supporters only for a subset of "suspicious" nodes; however, we do not know *a priori* which nodes are spammers. An efficient solution will be presented in Section 4.

3. RANK PROPAGATION

In this section we describe a link-based ranking method that produces a metric suitable for Web link spam detection.

Let $A_{N\times N}$ be the citation matrix of graph $G = (V, E)$, that is, $a_{xy} = 1 \iff (x, y) \in E$. Let $\mathbf{P}$ be the row-normalized version of the citation matrix, such that all rows sum up to one, and rows of zeros are replaced by rows of $1/N$ to avoid the effect of rank "sinks".

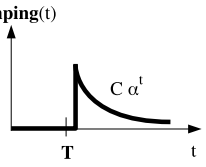
A **functional ranking** [1] is a link-based ranking algorithm to compute a scoring vector $\mathbf{W}$ of the form:

$$\mathbf{W} = \sum_{t=0}^{\infty} \frac{\text{damping}(t)}{N} \mathbf{P}^t \ .$$

where damping(t) is a decreasing function on t, the lengths of the paths. In particular, PageRank [22] is the most widely known functional ranking, in which the damping function is exponentially decreasing, namely, damping$(t) = (1-\alpha)\alpha^t$ where α is a damping factor between 0 and 1, typically 0.85.

A page participating in a link farm can gain a high PageRank score because it has many in-links, this is, supporters that are topologically "close" to the target node. Intuitively, a way of demoting those pages could be to consider a damping function that **ignores the direct contribution of the first levels of links**, such as:

$$\text{damping}(t) = \begin{cases} 0 & t \le T \\ C\alpha^t & t > T \end{cases}$$



Where C is a normalization constant and α is the damping factor used for PageRank. The normalization constant is such that $\sum_{t=0}^{\infty} \text{damping}(t) = 1$, so $C = (1-\alpha)/(\alpha^{T+1})$.

This function penalizes pages that obtain a large share of their PageRank from the first few levels of links; we call the corresponding functional ranking the **Truncated PageRank** of a page. This is similar to PageRank, except that the supporters that are too "close" to a target node, do not contribute towards its ranking.

Require: N: number of nodes, $0 < \alpha < 1$: damping factor, T≥ -1: distance for truncation
1: **for** i : 1 ... N **do** {Initialization}
2: R[i] $\leftarrow (1-\alpha)/((\alpha^{T+1})N)$
3: **if** T≥ 0 **then**
4: Score[i] $\leftarrow 0$
5: **else** {Calculate normal PageRank}
6: Score[i] $\leftarrow$ R[i]
7: **end if**
8: **end for**
9: distance = 1
10: **while not** converged **do**
11: Aux $\leftarrow$ **0**
12: **for** src : 1 ... N **do** {Follow links in the graph}
13: **for all** link from src to dest **do**
14: Aux[dest] $\leftarrow$ Aux[dest] + R[src]/outdegree(src)
15: **end for**
16: **end for**
17: **for** i : 1 ... N **do** {Apply damping factor α}
18: R[i] $\leftarrow$ Aux[i] $\times\alpha$
19: **if** distance > T **then** {Add to ranking value}
20: Score[i] $\leftarrow$ Score[i] + R[i]
21: **end if**
22: **end for**
23: distance = distance +1
24: **end while**
25: **return** Score

Figure 5: TruncatedPageRank Algorithm.

For calculating the Truncated PageRank, we use the following auxiliary variable:

$$\mathbf{R}^{(0)} = \frac{C}{N} \qquad \mathbf{R}^{(t)} = \alpha \mathbf{R}^{(t-1)} P \ ,$$

and we calculate the truncated PageRank by using:

$$\mathbf{W} = \sum_{t=T+1}^{\infty} \mathbf{R}^{(t)} \ .$$

The algorithm is presented in Figure 5. For the calculation, it is important to keep the score and the accumulator $\mathbf{R}^{(t)}$ separated in the calculation, since we discard the first levels, or we may end up with only zeros in the output. Note that, when $T = -1$, we compute the normal PageRank. In fact, writing $\mathbf{W}$ in closed form we have $\mathbf{W} = \frac{C}{N}(I - \alpha\,\mathbf{P})^{-1}(\alpha\,\mathbf{P})^{T+1}$ which shows an additional damping factor when $T > -1$.

When comparing the values obtained with PageRank with those of TruncatedPageRank in the `uk` graph, for values of T from 1 to 4. Figure 6 shows the result. As expected, both measures are highly correlated, and the correlation decreases as more levels are truncated.

We do not argue that Truncated PageRank should be used as a substitute for PageRank, but we show in section 5 that the ratio between Truncated PageRank and PageRank is extremely valuable for detecting link spam. In practice, for calculating the Truncated PageRank it is easier to save "snapshots" with the partial PageRank values obtained at an intermediate point of the computation, and then use those values to indirectly calculate the Truncated PageRank.

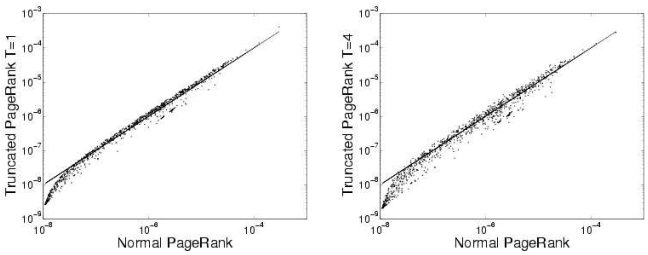


Figure 6: Comparing PageRank and Truncated PageRank with $T = 1$ and $T = 4$. Each dot represents a home page in the `uk` graph. The correlation is high and decreases as more levels are truncated.

4. ESTIMATION OF SUPPORTERS

In this section we describe a method for the estimation of the number of supporters of each node in the graph. Our method computes an estimation of the number of supporters for all vertices in parallel at the same time and can be viewed as a generalization of the ANF algorithm [23].

Since computing the number of supporters exactly is unfeasible on a large Web graph, we use probabilistic counting [6, 11]. As to this point, we propose a refinement of the classical probabilistic counting algorithm proposed in [11] and adopted in [23]. Our probabilistic counting algorithm is equivalent to the one proposed in [11], though it is more accurate when the distance under consideration is small, as is the case in the application we consider. As an algorithmic engineering contribution, our probabilistic counting algorithm is implemented with a generalization of the streaming algorithm used for PageRank computation [22, 17]. As a theoretical contribution, the probabilistic analysis of our base algorithm turns out to be considerably simpler than the one given in [11] for the original one.

4.1 General algorithm

We start by assigning a random vector of bits to each page. We then perform an iterative computation: on each iteration of the algorithm, if page y has a link to page x, then the bit vector of page x is updated as $x \leftarrow x$ `OR` y. In Figure 7, two iterations are shown. On each iteration, a bit set to `1` in any page can only move by one link in distance.

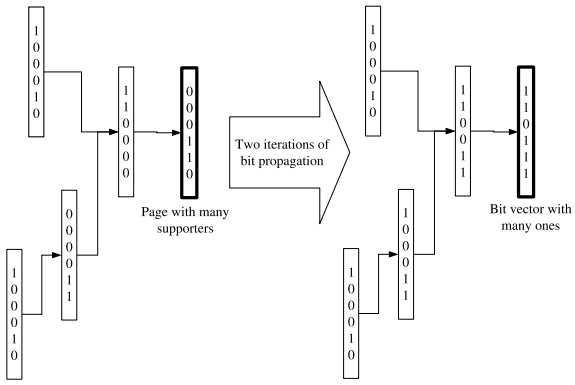


Figure 7: Schematic depiction of the bit propagation algorithm with two iterations.

```
Require: N: number of nodes, d: distance, k: bits
 1: for node : 1 ... N do {Every node}
 2:   for bit : 1 ... k do {Every bit}
 3:     INIT(node,bit)
 4:   end for
 5: end for
 6: for distance : 1...d do {Iteration step}
 7:   Aux ← 0_k
 8:   for src : 1 ... N do {Follow links in the graph}
 9:     for all links from src to dest do
10:       Aux[dest] ← Aux[dest] OR V[src,·]
11:     end for
12:   end for
13:   for node : 1 ... N do
14:     V[node,·] ← Aux[node]
15:   end for
16: end for
17: for node: 1...N do {Estimate supporters}
18:   Supporters[node] ← ESTIMATE( V[node,·] )
19: end for
20: return Supporters
```

Figure 8: Bit-Propagation Algorithm for estimating the number of distinct supporters at distance $\leq d$ of all the nodes in the graph simultaneously.

After d iterations, the bit vector associated to any page x provides information about the number of supporters of x at distance $\leq d$. Intuitively, if a page has a larger number of supporters than another, more 1s will appear in the final configuration of its bit vector. The algorithm, presented in Figure 8, can be efficiently implemented by using bit operations if k matches the word size of a particular machine architecture (e.g.: 32 or 64 bits).

The structure is the same as the algorithm in Figure 5, so the estimation of the number of supporters for all vertices in the graph can be computed concurrently with the execution of Truncated PageRank and PageRank.

The basic algorithm requires $O(kN)$ bits of memory, can operate over a streamed version of the link graph stored as an adjacency list, and requires to read the link graph d times. Its adaptive version, shown in Subsection 4.3 requires the same amount of memory and reads the graph $O(d \log N)$ times, while the backward variant presented in Subsection 4.4 reads the graph $O(d \log N_{\max}(d))$ times in the average, where $N_{\max}(d)$ is the maximum number of supporters at distance at most d, normally much smaller than N for the values of d that are of interest in our particular application.

Notation. Let $\mathbf{v}_i$ be the bit vector associated to any page, $i = 1, \ldots, N$. Let x denote a specific page and let $S(x, d)$ denote the set of supporters of this page within some given distance d. Let $N(x, d) = |S(x, d)|$. For concreteness, and according to Figure 3, we are considering typical values of d in the interval $1 \leq d \leq 20$. For the sake of simplicity, in the sequel we write $S(x)$ and $N(x)$ for $S(x, d)$ and $N(x, d)$ whenever we are considering a specific value of d.

4.2 Base estimation technique

INIT(node,bit): In the initialization step, the j-th bit of $\mathbf{v}_i$ is set to 1 with probability ϵ, independently for every $i = 1, \ldots, N$ and $j = 1, \ldots, k$ (ϵ is a parameter of the algorithm whose choice is explained below).

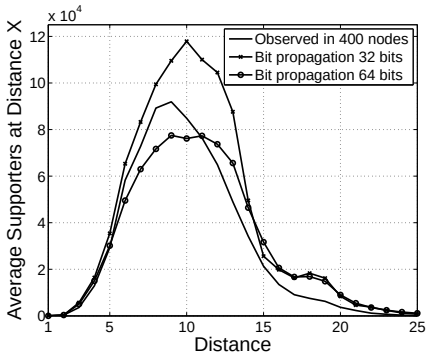


Figure 9: Comparison of the estimation of the average number of distinct supporters, against the observed value, in a sample of nodes.

Since ϵ is fixed, we can reduce the number of calls to the random number generator by generating a random number according to a geometric distribution with parameter $1 - \epsilon$ and then skipping a corresponding number of positions before setting a 1. This is especially useful when ϵ is small.

ESTIMATE(V[node,·]) Consider a page x, its bit vector $\mathbf{v}_x$ and let X_i be its i-th component, $i = 1, \ldots, k$. By the properties of the OR operator and by the independence of the X_i's we have,

$$\mathbf{P}[X_i = 1] \quad = \quad 1 - (1 - \epsilon)^{N(x)},$$

Also, if $B_\epsilon(x) = \sum_{i=1}^{k} X_i$, we have,

$$\mathbf{E}[B_\epsilon(x)] \quad = \quad [1 - (1 - \epsilon)^{N(x)}]\, k.$$

If we knew $\mathbf{E}[B_\epsilon(x)]$ we could compute $N(x)$ exactly. In practice, for every run of the algorithm and for every x we simply have an estimation $\overline{B}_\epsilon(x)$ of it. Our base estimator is:

$$\overline{N}(x) = \log_{(1-\epsilon)} \left(1 - \frac{\overline{B}_\epsilon(x)}{k}\right) .$$

In Figure 9 we show the result of applying the basic algorithm with $\epsilon = 1/N$ to the 860,000-nodes eu.int graph using 32 and 64 bits, compared to the observed distribution in a sample of nodes. It turns out that with these values of k, the approximation at least captures the behavior of the *average* number of neighbors. However, this is not good enough for our purposes, as we are interested in specific nodes. This motivates the next algorithm.

4.3 An adaptive estimator

The main problem with the basic technique is that, given some number k of bits to use, not all values of ϵ are likely to provide useful information. In particular, $N(x)$ can vary by orders of magnitudes as x varies. This means that for some values of ϵ, the computed value of $B_\epsilon(x)$ might be k (or 0, depending on $N(x)$) with relatively high probability. In order to circumvent this problem, we observe that, if we knew $N(x)$ and chose $\epsilon = 1/N(x)$ we would get:

$$\mathbf{E}[B_\epsilon(x)] \quad \simeq \quad \left(1 - \frac{1}{e}\right) k \simeq 0.63k,$$

where the approximation is very good for all values of $N(x, d)$ of interest. Also, $\mathbf{E}[B_\epsilon(x)]$ is a monotone function in ϵ and

we can reasonably expect to observe a transition of $\overline{B}_\epsilon(x)$ from a value smaller to a value larger than $(1-1/e)k$, as ϵ increases from values smaller to values larger than $1/N(x)$.

In practice, we apply the basic algorithm $O(\log(N))$ times as explained in Figure 10. The rough idea is as follows: starting with a value $\epsilon_{\min}$ (for instance, $\epsilon_{\min} = 1/N$) we proceed by doubling ϵ at each iteration, up to some value $\epsilon_{\max}$ (for instance, $\epsilon_{\max} = 0.5$). Given x, ϵ will at some point take up some value $\epsilon(x)$ such that $\epsilon(x) \leq 1/N(x) \leq 2\epsilon(x)$. Ideally, increasing ϵ from $\epsilon(x)$ to $2\epsilon(x)$, $B_\epsilon(x)$ should transit from a value smaller than $(1-1/e)k$ to a value larger than $(1-1/e)k$. This does not hold deterministically of course, but it holds with sufficiently high probability, if k is large enough.

Require: $\epsilon_{\min}$, $\epsilon_{\max}$ limits
1: $\epsilon \leftarrow \epsilon_{\min}$
2: **while** $\epsilon < \epsilon_{\max}$ and not all nodes have estimations **do**
3: Run the **Bit-Propagation algorithm** with ϵ
4: **for** x such that $B_\epsilon(x) > 0.63\,k$ *for the first time* **do**
5: Estimate $\overline{N}(x) \leftarrow 4/3\epsilon$
6: **end for**
7: $\epsilon \leftarrow \epsilon \times 2$
8: **end while**
9: **return** $\overline{N}(x)$

Figure 10: Adaptive Bit-Propagation algorithm for estimating the number of distinct supporters of all nodes in the graph. The algorithm calls the normal Bit-Propagation algorithm a number of times with varying values of ϵ.

LEMMA 1. *Algorithm* **Adaptive Bit-Propagation** *iterates for at most* $\log_2(\epsilon_{\max}/\epsilon_{\min}) = O(\log_2 N)$ *times.*

Note that we might have a "false positive", i.e. for some vertex x, it might be the case that $B_\epsilon(x) > (1-1/e)\,k$ for $\epsilon < \epsilon(x)$ or $B_\epsilon(x) < (1-1/e)\,k$ for $\epsilon > \epsilon(x)$. We prove that this is not likely to happen, provided k is large enough. The proof of this fact, based on the application of standard Chernoff's bounds [19], will appear in the full version of the paper.

Recall that $\overline{N}(x)$ denotes the estimation of $N(x)$ computed above and let $\overline{\epsilon} = 1/\overline{N}(x)$. Due to space limitations the proof of the following theorem will appear in the full version of the paper.

THEOREM 1. *Whenever* $0.018\,k \geq 1$, *for every* x:

$$\mathbf{P}\left[(\overline{N}(x) > 2N(x)) \cup \left(\overline{N}(x) < \frac{1}{2}N(x)\right)\right] \leq 2e^{-0.018\,k} + e^{-0.013\,k} + e^{-0.31\,k} + e^{-0.045\,k}.$$

We want to underline that the analysis of our method is much simpler than the one presented in [11]. The reason is that the probabilistic analysis in [11] requires computing the average position of the least significant bit that is not set to 1 in a suitably generated random bit string. Computing this value is not straightforward. Conversely, in our case, every $B_\epsilon(x)$ is the sum of binary independent random variables, so that we can easily compute its expectation and provide tight bounds to the probability that it deviates from the expectation for more than a given factor.

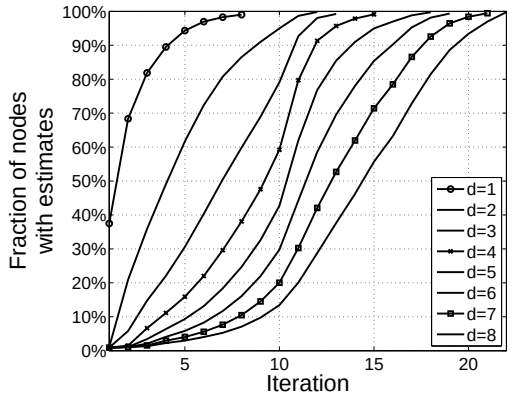


Figure 11: Fraction of nodes with good estimations after a certain number of iterations. For instance, when measuring at distance $d = 4$, 15 iterations suffice to have good estimators for 99% of the pages.

4.4 Experimental results

For the purpose of our experiments, we proceed backwards, starting with $\epsilon_{\max} = 1/2$ and then dividing ϵ by two at each iteration. This is faster than starting with a smaller value and then multiplying by two, mainly because in our case we are dealing with small distances and thus with neighborhoods in the order of hundreds or thousands of nodes. We freeze the estimation for a node when $B_\epsilon(x) < 0.63k$, and stop the iterations when 1% or less nodes have $B_\epsilon(x) \geq 0.63k$. Figure 11 shows that the number of iterations of the Adaptive Bit-Propagation algorithm required for estimating the neighbors at distance 4 or less is about 15, and for all distances up to 8 the number of iterations required is less than 25.

Besides the estimator described in the previous section, we considered the following one: whenever $B_\epsilon(x) < 0.63k$ for the first time, we estimate $\overline{N}(x)$ twice using the estimator from section 4.2 with $B_\epsilon(x)$ and $B_{2\epsilon}(x)$, and then average the resulting estimations. We call this the *combined* estimator that uses both the information from ϵ as well as the number of bits set. In practice the error of the combined estimator is lower.

We compared the precision obtained by this method with the precision given by the ANF algorithm [23]. In ANF, the size of the bitmask depends on the size of the graph, while the number of iterations (k in their notation) is used to achieve the desired precision. Our approach is orthogonal: the number of iterations depends on the graph size, while the size of the bitmask is used to achieve the desired precision. In order to compare the two algorithms fairly, we considered the product between the bitmask size and the number of iterations as a parameter describing the overall number of bits per node used (this is in particular the case if iterations are performed in parallel).

We fixed in ANF the size of the bitmask to 24, since $2^{24} = 16M$ is the closest number for the 18 million nodes of our collection (using more would be wasting bits). Next we ran ANF for 24 iterations (equivalent to 576 bits × iterations) and for 48 iterations (equivalent to 1152 bits × iterations). The former value is slightly more than the requirements of our algorithm at distance 1, while the latter is the same

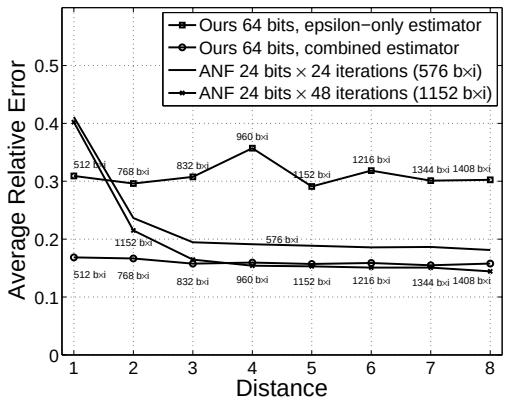


Figure 12: Comparison of the average relative error of the different strategies.

requirement as our algorithm at distance 5 (one plus the maximum distance we use for spam detection).

The comparison is shown in Figure 12. It turns out that the basic estimator performs well over the entire distance range, but both ANF and the combined estimator technique perform better. In particular, our combined estimator performs better than ANF for distances up to 5 (the ones we are interested in for the application we consider), in the sense that it has better average relative error and/or it has the same performance but it uses a smaller overall amount of bits. For larger distances the probabilistic counting technique used in ANF proves more efficient, since it has the same performance but it uses a smaller overall amount of bits.

It is also important to remark that, in practice, the memory allocation is in words of either 32 or 64 bits. This means that, even if we choose a bitmask of 24 bits for the ANF probabilistic counting routine, as was the case in our experiments, 32 bits are actually allocated to each node, 8 of which will not be used by the algorithm. With our approach instead, these bits can be used to increase precision. These considerations of efficiency are particularly important in the large data sets we are considering.

5. EXPERIMENTS ON SPAM DETECTION

One of the key issues in spam detection is to provide direct techniques that allow search engines to decide if a page can be trusted. We built a set of classifiers to test the efficiency of the metrics we have described for automatic spam classification.

5.1 Data set and sampling

Our collection is a set of 18.5 million pages from the `.uk` domain, downloaded in 2002. The pages were located in 98,452 different hosts. Given the large size of this collection, we decided to classify entire hosts instead of individual pages. While this introduces errors, as some hosts are a mixture of non-spam and spam content, it allows us to have much broader coverage. To provide class labels for the classifier, we manually inspected a sample of 5,750 hosts (5.9%) that covered 31.7% of the Web pages in the collection.

For every host, we inspected a few pages manually and looked at the list of URLs collected from that host by the Web crawler. Whenever we found a link farm inside the host, we classified the entire host as spam. In practice, in very few cases we observed mixtures of spam and non-spam content in the same host.

The manual classification stage took roughly three work days. As the amount of spam compared to normal pages is relatively small, and since we want to focus on the most "damaging" types of spam, we biased our sampling towards hosts with high PageRank. This is the same approach taken by other researchers in link-spam detection [5, 16]. We also biased our sampling towards hosts with a large number of pages and hosts with long names such as "`www.buy-a--used-car-today.example`". Neither the length of the host names nor the number of pages in the host were provided as attributes for the classification, as they are not link-based features.

When manually tagging, we discarded the hosts that were no longer available (about 7%) and classified the remaining 5,344 hosts into the following 3 categories:

Spam (16%): The host is clearly a link farm; or it is spamming by using unrelated keywords in the host, directory or file names; or it includes no content apart from links to a target page or host.

Normal (81%): The host is clearly not a link farm (in the jargon of e-mail spam detection tools, non-spam items are also called "ham").

Suspicious (3%): Borderline cases, including illegal business (on-line gambling, pharmaceuticals without prescription) and pornography, as they are usual customers of link farms. We also included in this category hosts that are almost entirely content copied from other sources plus advertising, affiliate networks, advertising servers, and groups of entire hosts that share the same template with little added information.

Table 1 shows the number of hosts and pages in each class.

Class	Hosts		Pages	
Spam	840	16%	329 K	6%
Normal	4,333	81%	5,429 K	92%
... Suspicious	171	3%	118 K	2%
Total	5,344	(5.8%)	5,877 K	(31.7%)

Table 1: Relative sizes of the classes in the manually-classified sample. The last row gives the fraction of classified hosts and pages over the entire collection.

There are many Web sites whose design is optimized for search engines, but also provide useful content. This is a gray area between "ethical" search engine optimization and "unethical" spamdexing. For the class labels provided to the algorithms in the automatic classification experiments, we adopted a conservative approach and included the suspicious pages in the normal class.

Also, we must note that the Web is a moving target, and it is possible that Web sites have changed from spam to non-spam or viceversa in the period from 2002 to date, so regardless of the technique used, this may introduce extra noise in the labels assigned to the hosts.

5.2 Automatic Classification

We extracted automatically a series of features from the data, including the PageRank, TruncatedPageRank at dis-

tance $d = 1, 2, 3$ and 4, and the estimation of supporters at the same distances, using the adaptive technique described in section 4. These link-based metrics are defined for pages, so we assigned them to hosts by measuring them at both the **home page** of the host (the URL corresponding to the root directory) and the page with the **maximum PageRank** of the host. In our sample, these pages are not the same in 62% of the cases, so it is rather common that the highest ranked page on a host is not the home page.

The classified hosts, grouped into the two manually-assigned class labels: "spam" and "normal" constitute the *training set* for the learning process. We experimented with the `Weka` [25] implementation of `C4.5` decision trees. Describing this classifiers here in detail in not possible due to space limitations, for a description see [25].

The evaluation of the learning schemes was performed by a ten-fold cross-validation of the training data. The data is first divided into 10 approximately equal partitions, then each part is held out in turn and the learning scheme is trained on the remaining 9 folds. The overall error estimate is the average of the 10 error estimates. The error metrics we are using are the precision and recall measures from information retrieval [3], considering the spam detection task:

$$\text{Precision} = \frac{\#\text{ of spam hosts classified as spam}}{\#\text{ of hosts classified as spam}}$$

$$\text{Recall} = \frac{\#\text{ of spam hosts classified as spam}}{\#\text{ of spam hosts}} .$$

And we measured the two types of errors of the spam classification:

$$\text{False positives} = \frac{\#\text{ of normal hosts classified as spam}}{\#\text{ of normal hosts}}$$

$$\text{False negatives} = \frac{\#\text{ of spam hosts classified as normal}}{\#\text{ of spam hosts}} .$$

The false negative rate is one minus the recall of the spam detection task, and the false positive rate is one minus the recall of the normal host detection task.

5.3 Combined classifier

We also included combinations of these features, including for instance: ratio of supporters at distance d versus supporters at distance $d-1$, ratio of Truncated PageRank at distance d versus PageRank, minimum ratio of neighbors at d and neighbors at $d-1$, and so on. In total, we used 82 features.

We built a complete decision tree that is post-pruned in order to reduce the number of rules generated. In addition we perform the reduced-error pruning step to optimize performance.

We ran the algorithm first with a minimum of 5 and 30 hosts per leaf (parameter M) of the tree leading respectively to a decision tree with 49 and 31 rules. We limit the rate of hosts per leaf to avoid overfitting, and to have a smaller set of rules that can be more easily studied and understood. The results are shown in table 2.

We also generated a decision tree without the reduced-error pruning step, that had 189 rules. This decision tree does not improve the performance compared with the tree of 49 rules, which detects 80% of the Web spam with 2% of error rate.

		Spam class		False	False
Pruning	Rules	Precision	Recall	Pos.	Neg.
$M = 5$	49	0.87	0.80	2.0%	20%
$M = 30$	31	0.88	0.76	1.8%	24%
No pruning	189	0.85	0.79	2.6%	21%

Table 2: Evaluation of the performance of classifiers based on a combination of Truncated PageRank and Estimation of Supporters. M is the minimum number of hosts per leaf in the decision tree.

5.4 Comparison

We implemented the TrustRank [16, 14] algorithm for testing it in our collection. TrustRank is a well-known algorithm for separating high-quality reputable pages/hosts from the rest, that is, spam and low-quality nodes. It provides a metric that can be used for ranking, and when combined with PageRank, can also be used for Web spam demotion.

The TrustRank calculation starts from a set of pages manually labeled as trusted, and does a random walk for a few steps (typically around 20), returning to the original set with a certain probability at each step (usually 0.15). The obtained probability of reaching a page is called its **estimated non-spam mass**. Pages with very low estimated non-spam mass compared to their PageRank should be considered suspicious. The intuition is that a page with a high PageRank, but with no connection with the most relevant pages on the Web, can be considered suspicious.

In our implementation, for the set of trusted nodes we used data from the Open Directory Project (Available at `http://dmoz.org/`). This is a manually built directory of Web sites that are considered to be of high quality. We considered as trusted the 32,866 hosts in our collection (from a total of 98,452) that had at least one Web page in the directory.

We build one automatic classifier for each of the three techniques we have described:

Classifier based on TrustRank: uses as features the PageRank, the estimated non-spam mass, and the estimated non-spam mass divided by PageRank.

Classifier based on Truncated PageRank: uses as features the PageRank, the Truncated PageRank with truncation distance $t = 2, 3, 4$ (with $t = 1$ it would be just based on in-degree), and the Truncated PageRank divided by PageRank.

Classifier based on Estimation of Supporters: uses as features the PageRank, the estimation of supporters at a given distance $d = 2, 3, 4$, and the estimation of supporters divided by PageRank.

For every classifier, we measured the attributes in both the home page and the page with the maximum PageRank of every host. In this way, all the classifiers receive as an input these six features plus the class label. Their accuracy after ten-fold cross-validation is presented in Table 3.

The best single-technique classifier is the one based in estimation of supporters, that can detect 57%-64% of the Web spam with 2.1%-2.0% of false positives. It is followed by the classifier based on TrustRank alone, that detects 49%-50%

Table 3: Comparison of single-technique classifiers.

Classifiers (pruning with $M = 5$)	Spam class Prec.	Recall	False Pos.	False Neg.
TrustRank	0.82	0.50	2.1%	50%
Trunc. PageRank $t = 2$	0.85	0.50	1.6%	50%
Trunc. PageRank $t = 3$	0.84	0.47	1.6%	53%
Trunc. PageRank $t = 4$	0.79	0.45	2.2%	55%
Est. Supporters $d = 2$	0.78	0.60	3.2%	40%
Est. Supporters $d = 3$	0.83	0.64	2.4%	36%
Est. Supporters $d = 4$	**0.86**	**0.64**	**2.0%**	**36%**

Classifiers (pruning with $M = 30$)	Spam class Prec.	Recall	False Pos.	False Neg.
TrustRank	0.80	0.49	2.3%	51%
Trunc. PageRank $t = 2$	0.82	0.43	1.8%	57%
Trunc. PageRank $t = 3$	0.81	0.42	1.8%	58%
Trunc. PageRank $t = 4$	0.77	0.43	2.4%	57%
Est. Supporters $d = 2$	0.76	0.52	3.1%	48%
Est. Supporters $d = 3$	**0.83**	**0.57**	**2.1%**	**43%**
Est. Supporters $d = 4$	0.80	0.57	2.6%	43%

of the Web spam with 2.3%-2.1% of error and the classifier based on Truncated PageRank, which detects 43%-50% of the Web spam with 1.8%-1.6% of false positives.

None of these single-technique classifier is as good as the combined classifier shown in Table 2. A comparison of the performance of these classifiers is shown in Figure 13. A full comparison of all the combinations is outside the scope of this paper; in [4] we further study these and other metrics for link-based Web spam detection, and combine several of them to produce Web spam classifiers.

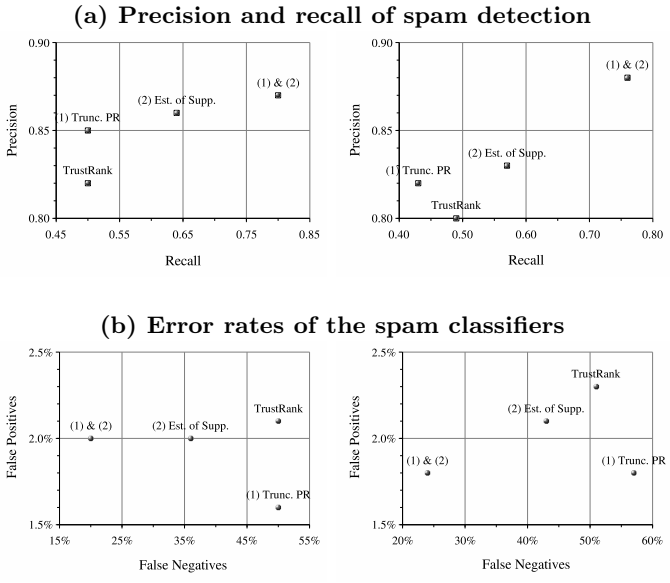


Figure 13: Comparison of single-technique classifiers and a combined classifier using the algorithms described on this paper. Left: pruning with $M = 5$, right: pruning with $M = 30$.

6. RELATED WORK

This section discusses previous work in the areas of link spam detection, propagation of trust, and probabilistic counting.

Detecting spam: In [10] it is shown that most outliers in the histograms of certain properties of Web pages (such as in-degree and out-degree) are groups of spam pages. The method of "shingles" for detecting dense sub-graphs [12] can be also applied for link farm detection, as members of a link farm may share a substantial fraction of their out-links (however, the algorithm will perform worse if the link farm is randomized).

In [27] it is shown that spam pages should be very sensitive to changes in the damping factor of the PageRank calculation; in our case with Truncated PageRank we modify not only the damping factor but the whole damping function.

In [5] a different approach for detecting link spam is proposed. They start from a suspicious page, follow links backwards to find pages which are strong contributors of PageRank for the target node, and then measure if the distribution of *their* PageRank is a power-law or they are mostly pages in a narrow PageRank interval. Note that this can only be done for some pages at the same time, while all the algorithms we apply can be executed *for all nodes in the graph at the same time.*

Also, content-based analysis [21, 8, 7] has been used for detecting spam pages, by studying relevant features such as page size or distribution of keywords, over a manually tagged set of pages. Content-based analysis is orthogonal to our approach, and thus both techniques can be combined to increase the performance of spam detection in a real search engine.

The performance of content-based classification is comparable to our approach. A state-of-the-art content-based classifier described in [21], without bagging nor boosting (equivalent to the classifiers we have presented here), reported 82% of recall, with 2.5% of false positives. With our classifier we had 80% of recall with 2.0% of false positives. Unfortunately, their classifier is not publicly available for evaluating its performance in the same collection as ours. It is likely that Web spam classifiers will be kept as business secrets by most researchers related to search engines, and this implies that for evaluation it will be necessary to have a common reference collection for the task of Web spam detection in general.

Propagating trust and "spamicity": It is important to notice that we do not need to detect all spam pages, as the "spamicity" can be propagated. A technique shown in [26] is based on finding a page that is part of a link farm and then marking all pages that link to it, possibly marking recursively following back-links up to a certain threshold (this is also called "BadRank").

In [5], "spamicity" is propagated by running a personalized PageRank in which the personalization vector demotes pages that are found to be spam.

Probabilistic counting: Morris' algorithm (1978) [20] was the first randomized algorithm for counting up to a large number with a few bits. A more sophisticated technique for probabilistic counting is presented in [11]; this technique is applied to the particular case of counting the number of in-neighbors or "supporters" of a page in [23]. The use of probabilistic counting is important in this case, as the cost of calculating the exact values is prohibitive [18].

7. CONCLUSIONS AND FUTURE WORK

In this paper we have shown how two different algorithms: rank propagation and probabilistic counting, can be viewed as instances of a PageRank-like computation. We have applied these algorithms for Web spam detection and shown that they can improve state-of the art Web spam detection techniques.

Most modern search engines use a combination of factors for ranking search results. These factors include information aggregated from different sources, such as user clicks, text similarity and link analysis. The "spamicity" of a page is another factor and we must find how to combine it with the other sources of information.

If we can estimate the probability of a page being spam, then a natural combination could be to rank the pages according to the product of their score and the probability of a page being not spam –if we are 100% sure that a page is spam, then its final score will be zero.

The performance of web spam detection algorithms in general (not only ours) is still modest when compared with the error rate of modern e-mail spam detection systems. Fortunately, web spam detection can be more strict than e-mail spam detection. While losing a relevant e-mail message is very bad, demoting a relevant Web page in the search results is not so bad, because if the page is relevant, it can be found later by following links, or it can be moved to the second or third page of results.

However, the current precision and recall of Web spam detection algorithms can be improved. An ambitious goal we propose is to try to achieve the same precision and recall of e-mail spam detection algorithms. In our opinion, this is unfeasible with link-only or content-only classifiers, and requires the combination of both methods. It also requires assembling a common reference collection of Web spam, and we intend to work with other researchers towards this goal. Nevertheless, our technique does not need any previous spamicity information to work.

The source code of the Java implementation of the algorithms presented in this paper will be freely available under a GPL license at `http://www.dis.uniroma1.it/~ae/` for the final version of the paper. The class labels we manually assigned will be available at the same address for repeatability of these results and further testing of these and other web spam detection techniques.

8. REFERENCES

[1] R. Baeza-Yates, P. Boldi, and C. Castillo. Generalizing PageRank: Damping functions for link-based ranking algorithms. In *Proceedings of SIGIR*, Seattle, Washington, USA, August 2006. ACM Press.

[2] R. Baeza-Yates, C. Castillo, and V. López. Pagerank increase under different collusion topologies. In *First International Workshop on Adversarial Information Retrieval on the Web*, 2005.

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

[4] L. Becchetti, C. Castillo, D. Donato, S. Leonardi, and R. Baeza-Yates. Link-based characterization and detection of web spam. Technical report, AEOLUS/Università di Roma La Sapienza, 2006.

[5] A. A. Benczúr, K. Csalogány, T. Sarlós, and M. Uher. Spamrank: fully automatic link spam detection. In *Proceedings of the First International Workshop on Adversarial Information Retrieval on the Web*, Chiba, Japan, May 2005.

[6] E. Cohen. Size-estimation framework with applications to transitive closure and reachability. *Journal of Computer and System Sciences*, 55(3):441–453, December 1997.

[7] B. D. Davison. Recognizing nepotistic links on the web. In *Aaai-2000 Workshop On Artificial Intelligence For Web Search*, pages 23–28, Austin, Texas, July 2000. Aaai Press.

[8] I. Drost and T. Scheffer. Thwarting the nigritude ultramarine: learning to identify link spam. In *Proceedings of the 16th European Conference on Machine Learning (ECML)*, volume 3720 of *Lecture Notes in Artificial Intelligence*, pages 233–243, Porto, Portugal, 2005.

[9] N. Eiron, K. S. Curley, and J. A. Tomlin. Ranking the web frontier. In *Proceedings of the 13th international conference on World Wide Web*, pages 309–318, New York, NY, USA, 2004. ACM Press.

[10] D. Fetterly, M. Manasse, and M. Najork. Spam, damn spam, and statistics: Using statistical analysis to locate spam web pages. In *Proceedings of the seventh workshop on the Web and databases (WebDB)*, pages 1–6, Paris, France, June 2004.

[11] P. Flajolet and N. G. Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182–209, 1985.

[12] D. Gibson, R. Kumar, and A. Tomkins. Discovering large dense subgraphs in massive graphs. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 721–732. VLDB Endowment, 2005.

[13] A. Gulli and A. Signorini. The indexable Web is more than 11.5 billion pages. In *Poster proceedings of the 14th international conference on World Wide Web*, pages 902–903, Chiba, Japan, 2005. ACM Press.

[14] Z. Gyöngyi, P. Berkhin, H. G. Molina, and J. Pedersen. Link spam detection based on mass estimation. Technical report, Stanford University, California, 2005.

[15] Z. Gyöngyi and H. Garcia-Molina. Web spam taxonomy. In *First International Workshop on Adversarial Information Retrieval on the Web*, 2005.

[16] Z. Gyöngyi, H. G. Molina, and J. Pedersen. Combating web spam with trustrank. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases (VLDB)*, pages 576–587, Toronto, Canada, August 2004. Morgan Kaufmann.

[17] T. Haveliwala. Efficient computation of pagerank. Technical report, Stanford University, 1999.

[18] R. J. Lipton and J. F. Naughton. Estimating the size of generalized transitive closures. In *VLDB '89: Proceedings of the 15th international conference on Very large data bases*, pages 165–171, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[19] M. Mitzenmacher and E. Upfal. *Probability and Computing : Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, January 2005.

[20] R. Morris. Counting large numbers of events in small registers. *Commun. ACM*, 21(10):840–842, October 1978.

[21] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly. Detecting spam web pages through content analysis. In *Proceedings of the World Wide Web conference*, pages 83–92, Edinburgh, Scotland, May 2006.

[22] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: bringing order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

[23] C. R. Palmer, P. B. Gibbons, and C. Faloutsos. ANF: a fast and scalable tool for data mining in massive graphs. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 81–90, New York, NY, USA, 2002. ACM Press.

[24] A. Perkins. The classification of search engine spam. Available online at http://www.silverdisc.co.uk/articles/spam-classification/, September 2001.

[25] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, October 1999.

[26] B. Wu and B. D. Davison. Identifying link farm spam pages. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 820–829, New York, NY, USA, 2005. ACM Press.

[27] H. Zhang, A. Goel, R. Govindan, K. Mason, and B. Van Roy. Making eigenvector-based reputation systems robust to collusion. In *Proceedings of the third Workshop on Web Graphs (WAW)*, volume 3243 of *Lecture Notes in Computer Science*, pages 92–104, Rome, Italy, October 2004. Springer.