Link Analysis for Web Spam Detection

LUCA BECCHETTI [1]
CARLOS CASTILLO [1,2]
DEBORA DONATO [1,2]
RICARDO BAEZA-YATES [2]
STEFANO LEONARDI [1]

[1] Università di Roma "La Sapienza"
[2] Yahoo! Research, Barcelona

We propose link-based techniques for automating the detection of Web spam, a term referring to pages which use deceptive techniques to obtain undeservedly high scores in search engines. The issue of Web spam is widespread and difficult to solve, mostly due to the large size of the Web which means that, in practice, many algorithms are infeasible.

We perform a statistical analysis of a large collection of Web pages. In particular, we compute statistics of the links in the vicinity of every Web page applying rank propagation and probabilistic counting over the entire Web graph in a scalable way. We build several automatic web spam classifiers using different techniques. This paper presents a study of the performance of each of these classifiers alone, as well as their combined performance.

Based on these results we propose spam detection techniques which only consider the link structure of Web, regardless of page contents. These statistical features are used to build a classifier that is tested over a large collection of Web link spam. After ten-fold cross-validation, our best classifiers have a performance comparable to that of state-of-the-art spam classifiers that use content attributes, and orthogonal to their methods.

Categories and Subject Descriptors: H.3.3 [**Information Systems**]: Information Search and Retrieval

General Terms: Algorithms, Measurement

Additional Key Words and Phrases: Link analysis, Adversarial Information Retrieval

Author's address: Via Salaria 113, Second Floor, Rome 00198. Rome, ITALY.

Preliminary results of this research were presented during the AIRWeb 2006 workshop [Becchetti et al. 2006a] and the WebKDD 2006 workshop [Becchetti et al. 2006b].
Partially supported by EU Integrated Project IST-015964 AEOLUS.
This is a preprint of a work in progress. Contact the authors for citation information.
Last updated: March 22, 2007

Contents

Last updated: March 22, 2007.

1. INTRODUCTION

The Web is 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 information. Search engines are the "dragons" that keep a valuable treasure: information [Gori and Witten 2005]. 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 that are attracted by the prospect of reaching millions of users at a very low cost. A large fraction of the visits to a Web site originate from search engines, and most of the users click on the first few results in a search engine. Therefore, there is an economic incentive for manipulating search engine's listings by creating pages that score high independently of their real merit. In practice such manipulation is widespread, and in many cases, successful. For instance, the authors of [Eiron et al. 2004] 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".

The term "**spam**" has been commonly used in the Internet era to refer to *unsolicited (and possibly commercial) bulk messages*. The most common form of electronic spam is **e-mail spam**, but in practice each new communication medium has created a new opportunity for sending unsolicited messages. These days there are many types of electronic spam, including spam by instant messaging (*spim*), spam by internet telephony (*spit*), spam by mobile phone, by fax, etc. The Web is not absent from this list, but as the request-response paradigm of the HTTP protocol makes it impossible for spammers to actually "send" pages directly to the users, spammers try to deceive search engines and thus break the trust that search engines establish with their users.

1.1 What is Web spam?

All deceptive actions which try to increase the ranking of a page in search engines are generally referred to as **Web spam** or **spamdexing** (a *portmanteau*, or combination, of "spam" and "index"). A **spam page or host** is a page or host that is either used for spamming or receives a substantial amount of its score from other spam pages.

An alternative way of defining Web spam could be any attempt to get "an unjustifiably favorable relevance or importance score for some web page, considering the page's true value" [Gyöngyi and Garcia-Molina 2005]. A **spam page** is a page which is used for spamming or receives a substantial amount of its score from other spam pages. Another definition of spam, given in [Perkins 2001] 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".

Seeing as there are many steps which content providers can take to improve the ranking of their Web sites, and given that is an important subjective element in the evaluation of the relevance of Web pages, to offer an exact definition of Web spam would be misleading. Indeed, there is a large gray area between "ethical" **Search Engine Optimization** (SEO) services and "unethical" spam. SEO services range from ensuring that Web pages are indexable by Web crawlers, to the creation of thousands or millions of fake pages aimed at deceiving search engine ranking algorithms. Our main criteria for deciding in borderline cases is the perceived effort spent by Web authors on providing good content, against the effort spent on trying to score highly in search engines.

The relationship between a Web site administrator trying to rank high in a search engine and the search engine administrator is an **adversarial** relationship in a zero-sum game. Each time a web site makes an unmerited gain in ranking, the accuracy of the search engine is reduced. However, more than one form of Web spam exists which involves search engines. For example, we do not take advertising spam into consideration, an issue which also affects search engines through fraudulent clicking on advertising.

1.2 Link-based Web spam (topological spam)

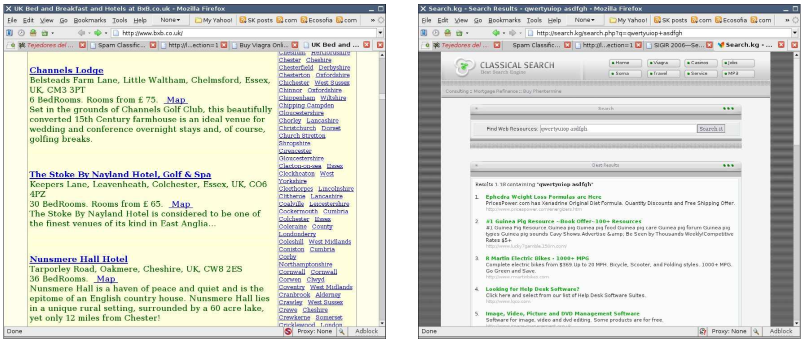
There are many techniques for Web spam [Gyöngyi and Garcia-Molina 2005], and they can be broadly classified in two groups: content (or keyword) spam, and link spam.

Content spam refers to changes in the content of the pages, for instance by inserting a large number of keywords [Davison 2000a; Drost and Scheffer 2005]. In [Ntoulas et al. 2006], it is shown that 82-86% of spam pages of this type can be detected by an automatic classifier. The features used for the classification include, amongst others: the number of words in the text of the page, the number of hyperlinks, the number of words in the title of the pages, the compressibility (redundancy) of the content, etc.

Link spam includes changes to the link structure of the sites, by creating **link farms** [Zhang et al. 2004; Baeza-Yates et al. 2005]. A link farm is a densely connected set of pages, created explicitly with the purpose of deceiving a link-based ranking algorithm. Zhang et. al. [Zhang et al. 2004] define this form of collusion 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". The pages in Figure 1 are part of **link farms**.

The targets of our spam-detection algorithms are the pages that receive most of their link-based 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, we show a schematic diagram depicting the links around a spam page and a normal page. Link farms can receive links from non-spam sites by buying advertising, or by buying expired domains used previously for legitimate purposes.

A page that participates in a link farm, such as the one depicted in Figure 2, may have a high in-degree, but little relationship with the rest of the graph. Heuristically, we refer to spamming achieved by using link farms as **topological spamming**. In particular, a topological spammer achieves its goal by means of a link farm that has topological and spectral properties that statistically differ from those exhibited by non spam pages. This definition embraces the cases considered in [Gibson et al.


UK Bed and Breakfast and Hotels at BxB.co.uk - Mozilla Firefox
File Edit View Go Bookmarks Tools Help None My Yahoo! SK posts com Ecosofia com
http://www.bxb.co.uk/
Tejedores del ... Spam Classific... http://l...ection=1 Buy Viagra Onli... UK Bed and ...
Done Proxy: None Adblock

Channels Lodge
Belsteads Farm Lane, Little Waltham, Chelmsford, Essex, UK, CM3 3PT
6 BedRooms. Rooms from £ 75. Map
Set in the grounds of Channels Golf Club, this beautifully converted 15th Century farmhouse is an ideal venue for wedding and conference overnight stays and, of course, golfing breaks.

The Stoke By Nayland Hotel, Golf & Spa
Keepers Lane, Leavenheath, Colchester, Essex, UK, CO6 4PZ
30 BedRooms. Rooms from £ 65. Map
The Stoke By Nayland Hotel is considered to be one of the finest venues of its kind in East Anglia...

Nunsmere Hall Hotel
Tarporley Road, Oakmere, Cheshire, UK, CW8 2ES
36 BedRooms. Map
Nunsmere Hall is a haven of peace and quiet and is the epitome of an English country house. Nunsmere Hall lies in a unique rural setting, surrounded by a 60 acre lake, yet only 12 miles from Chester!

Cheshunt Hertfordshire
Chester Cheshire
Chesterfield Derbyshire
Chesterton Oxfordshire
Chichester West Sussex
Chinnor Oxfordshire
Chippenham Wiltshire
Chipping Campden Gloucestershire
Chorley Lancashire
Christchurch Dorset
Church Stretton Shropshire
Cirencester Gloucestershire
Clacton-on-sea Essex
Cleckheaton West Yorkshire
Cleethorpes Lincolnshire
Clitheroe Lancashire
Coalville Leicestershire
Cockermouth Cumbria
Colchester Essex
Coleraine County Londonderry
Coleshill West Midlands
Coniston Cumbria
Corby Northamptonshire
Cornwall Cornwall
Corwen Clwyd
Coventry West Midlands
Cranbrook Alderney
Crawley West Sussex
Crewe Cheshire
Crewkerne Somerset
Cricklewood London

Search.kg - Search Results - qwertyuiop asdfgh - Mozilla Firefox
File Edit View Go Bookmarks Tools Help None My Yahoo! SK posts com Ecosofia com
http://search.kg/search.php?q=qwertyuiop+asdfgh
Tejedores del ... Spam Classific... http://l...ection=1 SIGIR 2006—Se... Search.kg - ...
CLASSICAL SEARCH
Best Search Engine
Home Viagra Casinos Jobs
Soma Travel Service MP3
Consulting :: Mortgage Refinance :: Buy Phentermine
Search
Search it
Best Results
Done Proxy: None Adblock

Find Web Resources: qwertyuiop asdfgh

Results 1-18 containing "**qwertyuiop asdfgh**"

1. **Ephedra Weight Loss Formulas are Here**
   PricesPower.com has Xenadrine Original Diet Formula. Quantity Discounts and Free Shipping Offer.
   http://www.pricespower.com/energizers.htm
2. **#1 Guinea Pig Resource ~Book Offer~100+ Resources**
   #1 Guinea Pig Resource.Guinea pig Guinea pig food Guinea pig care Guinea pig forum Guinea pig types Guinea pig sounds Cavy Shows.Advertise &amp; Be Seen by Thousands Weekly!Competitive Rates $5+
   http://www.lucky7gamble.150m.com/
3. **R Martin Electric Bikes - 1000+ MPG**
   Complete electric bikes from $369.Up to 20 MPH. Bicycle, Scooter, and Folding styles. 1000+ MPG. Go Green and Save.
   http://www.rmartinbikes.com
4. **Looking for Help Desk Software?**
   Click here and select from our list of Help Desk Software Suites.
   http://www.lqco.com
5. **Image, Video, Picture and DVD Management Software**
   Software for image, video and dvd editing. Some products are for free.
   http://www.image-management.org.uk

Fig. 1. Examples of Web spam pages belonging to link farms. While the page on the left has content features that can help to identify it as a spam page, the page on the right looks more similar to a "normal" page and thus can be more easily detected by its link attributes.

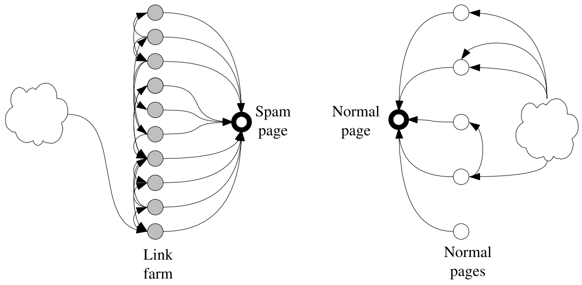


Fig. 2. Schematic depiction of the neighborhood of a page participating in a link farm (left) and a normal page (right). A link farm is a densely connected sub-graph, with little relationship with the rest of the Web, but not necessarily disconnected.

2005], and their method based on "shingles" can be also applied in detecting some types of link farms (those that are dense graphs).

Link-based and content-based analysis offer two orthogonal approaches. We do not believe that these approaches are alternatives, on the contrary, they must be used together.

On one hand, in fact, link-based analysis does not capture all possible cases of spamming, since some spam pages appear to have spectral and topological properties that are statistically close to those exhibited by non spam pages. In this case, content-based analysis can prove extremely useful.

On the other hand, content-based analysis seems less resilient to changes in spammers strategies, in much the same way that content-based techniques for detecting email spamming are. For instance, a spammer could copy an entire Web site (creating a set of pages that may be able to pass all tests for content spam detection) and change a few out-links in every page to point to the target page. This may be a relatively inexpensive task to perform in an automatic way, whereas creating, maintaining, reorganizing a link farm, possibly spanning more than one domain, is economically more expensive.

It is important to note that there are some types of Web spam which are not completely link-based, and it is very likely that there are some hybrid structures which combine both link farms (for achieving a high link-based score) and content-based spam, having a few links, to avoid detection. In our opinion, the approach whereby content features, link-based features and user interaction (e.g.: data collected via a toolbar or by observing clicks in search engine results) are mixed, should work better in practice than a pure link-based method. In this paper, we focus on detecting link farms, since they seem to be an important ingredient of current spam activity.

A Web search engine operator must consider that "any evaluation strategy which counts replicable features of Web pages is prone to manipulation" [Page et al. 1998]. Fortunately, from the search engine's perspective, "victory does not require perfection, just a rate of detection that alters the economic balance for a would-be spammer" [Ntoulas et al. 2006].

1.3 Our contributions

Fetterly et al. hypothesized that studying the distribution of statistics about pages could be a good way of detecting spam pages, as "in a number of these distributions, outlier values are associated with web spam" [Fetterly et al. 2004]. The approach of this paper is to use link-based statistics and apply them to create classifiers suitable for Web spam detection. We include both algorithmic and experimental contributions.

On the algorithmic side, we adapt two link-based algorithms which tackle the issue of web spam detection. We introduce a damping function for rank propagation [Baeza-Yates et al. 2006] that provides a metric that helps in separating spam from non-spam pages. Then, we propose an approximate counting technique that can be easily "embedded" within a rank computation. This sheds new light upon, and simplifies the method proposed in [Palmer et al. 2002], suggesting that the base rank propagation algorithm provides a general framework for computing several relevant metrics. These algorithms were described in preliminary form in [Becchetti et al. 2006b], here we provide bounds on their running time and error rate.

On the experimental side, we describe an automatic classifier that only uses link-based attributes, without looking at Web page content, still achieving a precision that is comparable to that of the best spam classifiers that use content analysis. This is an important point, since in many cases spam pages exhibit contents that look "normal". Experimental results over a collection tagged by only one person (one of the authors of this paper) were presented in [Becchetti et al. 2006a], here we present experimental results over a larger collection tagged by over 20 volunteers [Castillo et al. 2006].

2. ALGORITHMIC FRAMEWORK

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 **simultaneously** compute 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.

We view our set of Web pages as a **Web graph**, that 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 $N = |V|$ in the full Web indexable by search engines is in the order of 10^{10} [Gulli and Signorini 2005], and the typical number of links per Web page is between 20 and 30.

2.1 Supporters

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 [Benczúr et al. 2005], 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.

In Figure 3 we plot the distribution of distinct supporters for a random sample of 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/`).

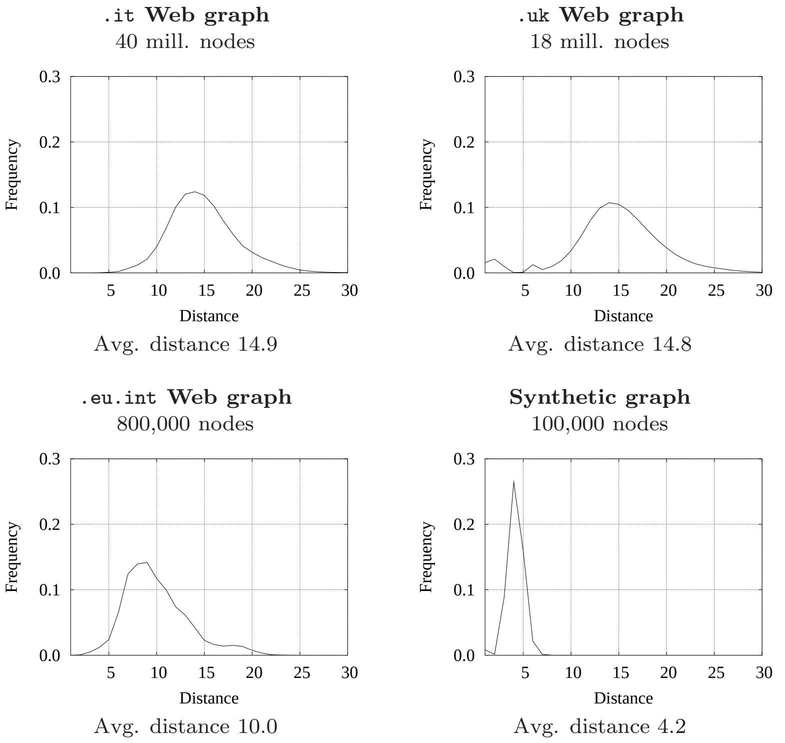
As suggested by Figure 2, a particular characteristic of a link farm is that the spam pages might have a large number of distinct supporters at short distances, but this number should be lower than expected at higher distances.

We can see that the number of new distinct supporters increases up to a certain distance, 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. In order to put this theory into practice, we calculated the PageRank of the pages in the `eu.int` (European Union) sub-domain. We chose this domain because it is a large, 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 will be 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, to compute this distribution for all the nodes in a large Web graph is computationally very expensive. A straightforward approach is to repeat a reverse breadth-first search from each node of the graph, and marking nodes as they are visited [Lipton and Naughton 1989]; the problem is that this would require $\Omega(N^2)$ memory for the marks if done in parallel or $\Omega(N^2)$ time to repeat a BFS from each one of the N


.it **Web graph**
40 mill. nodes

Avg. distance 14.9

.uk **Web graph**
18 mill. nodes

Avg. distance 14.8

.eu.int **Web graph**
800,000 nodes

Avg. distance 10.0

Synthetic graph
100,000 nodes

Avg. distance 4.2

Fig. 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 four large Web graphs [Baeza-Yates et al. 2006].

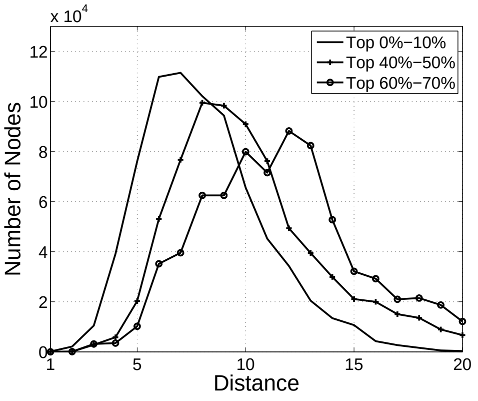


Fig. 4. Distribution of the number of new supporters at different distances, for pages in different PageRank buckets.

nodes if done sequentially. A possible solution could be to compute the supporters only for a subset of "suspicious" nodes; the problem with this approach is that we do not know *a priori* which nodes are spammers. An efficient solution will be presented in Section 4.

2.2 Semi-streaming graph algorithms

Given the large/huge size of typical data sets used in Web Information Retrieval, complexity issues are very important. This imposes severe restrictions on the computational and/or space complexity of viable algorithmic solutions. A first approach to modeling these restrictions could be the **streaming model** of computation [Henzinger et al. 1999]. However, the restrictions of the classical stream model are too severe and are hardly compatible with the problems we are interested in.

In light of the above remarks, we decided to restrict to algorithmic solutions whose space and time complexity is compatible with the **semi-streaming** model of computation [Feigenbaum et al. 2004; Demetrescu et al. 2006]. This implies a semi-external memory constraint [Vitter 2001] and thus reflects many significant constraints arising in practice. In this model, the graph is stored on disk as an adjacency list and no random access is possible, i.e., we only allow sequential access. Every computation involves a limited number of sequential scans of data stored in secondary memory [Haveliwala 1999].

Our algorithms also use an amount of main memory in the order of the number of nodes, whereas an amount of memory in the order of the number of edges may not be feasible. We assume that we have $\mathrm{O}(N \log N)$ bits of main (random access) memory, i.e., in general there is enough memory to store some limited amount of data about each vertex, but not to store the links of the graph in main memory. We impose a further constraint, i.e., the algorithm should perform a small number of passes over the stream data, at most $\mathrm{O}(\log N)$.

We assume no previous knowledge about the graph, so we do not know *a priori* if a particular node is suspicious of being a spam or not. For this reason, there are some semi-streamed algorithms on a Web graph that we cannot use for Web spam detection in our framework. If we have to compute a metric which assigns a value to every vertex, e.g. a score, we obviously cannot afford to run this algorithm again for every node in the graph, due to the large size of the data set.

As an example, suppose we want to measure the centrality of nodes. If we use the streamed version of the standard breadth-first search (BFS) algorithm, we are not complying to this requirement, since the outcome would be a BFS tree for a specific node, which is not enough for computing the centrality of all the nodes in the graph. Conversely, an algorithm such as PageRank computes a score for all nodes in the graph at the same time.

The general sketch of the type of semi-streamed graph algorithms we are interested, is shown in Figure 5.

According to Figure 5, we initialize a vector S that will contain some metric and possibly also auxiliary information. The size of S, $|S|$ is $\mathrm{O}(N)$. Then we scan the graph sequentially updating S according to observations on the graph. Then we post-process S and start over. This algorithmic sketch essentially captures all feasible algorithms on a large graph.

```
Require: graph $G = (V, E)$, score vector $S$
INITIALIZE($S$)
while not CONVERGED do
    for $src : 1 \ldots |V|$ do
        for all links from $src$ to $dest$ do
            COMPUTE($S, src, dest$)
        end for
    end for
    POST_PROCESS($S$)
end while
return $S$
```

Fig. 5. Generic link-analysis algorithm using a stream model. The score vector S represents any metric, and it must use $\mathrm{O}(N \log N)$ bits. The number of iterations should be $\mathrm{O}(\log N)$ in the worst case.

3. TRUNCATED PAGERANK

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".

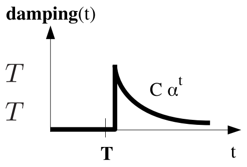
A **functional ranking** [Baeza-Yates et al. 2006] is a link-based ranking algorithm to compute a scoring vector $\mathbf{W}$ of the form:

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

where damping(t) is a decreasing function on t, the lengths of the paths. In particular, PageRank [Page et al. 1998] is the most widely known functional ranking, in which the damping function is exponentially decreasing, namely, $\text{damping}(t) = (1 - \alpha)\alpha^t$ 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, that is, supporters that are topologically "close" to the target node. Intuitively, a possible 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:

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



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**

```
Require: N: number of nodes, $0 < \alpha < 1$: damping factor, T$\geq -1$: distance for truncation
for i : 1 ... N do {Initialization}
    R[i] $\leftarrow (1-\alpha)/((\alpha^{T+1})N)$
    if T$\geq$ 0 then
        Score[i] $\leftarrow$ 0
    else {Calculate normal PageRank}
        Score[i] $\leftarrow$ R[i]
    end if
end for
distance = 1
while not converged do
    Aux $\leftarrow \mathbf{0}$
    for src : 1 ... N do {Follow links in the graph}
        for all link from src to dest do
            Aux[dest] $\leftarrow$ Aux[dest] + R[src]/outdegree(src)
        end for
    end for
    for i : 1 ... N do {Apply damping factor $\alpha$}
        R[i] $\leftarrow$ Aux[i] $\times\alpha$
        if distance > T then {Add to ranking value}
            Score[i] $\leftarrow$ Score[i] + R[i]
        end if
    end for
    distance = distance +1
end while
return Score
```

Fig. 6. **TruncatedPageRank Algorithm**.

PageRank of a page. This is similar to PageRank, except that supporters that are too "close" to a target node do not contribute to its ranking.

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

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

and we compute the truncated PageRank by using:

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

The algorithm is presented in Figure 6 and follows the general algorithmic sketch of 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 $T > -1$.

We compared the values obtained with PageRank with those of TruncatedPageRank in the `UK-2002` dataset, for values of T from 1 to 4. Figure 7 shows the result. As expected, both measures are closely correlated, and the correlation decreases as more levels are truncated.

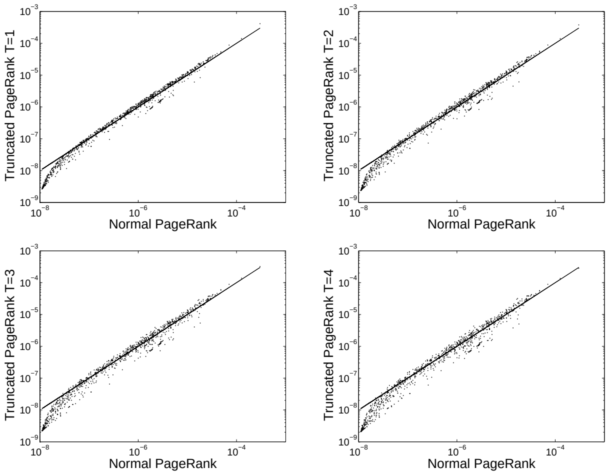


Fig. 7. Comparing PageRank and Truncated PageRank with $T = 1$ and $T = 4$. Each dot represents a home page in the `uk-2002` graph. The correlation is high and decreases as more levels are truncated.

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. Thus, computing Truncated PageRank has no extra cost (in terms of processing) for a search engine if the search engine already computes the PageRank vector for the Web graph.

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 [Palmer et al. 2002].

Since exactly computing the number of supporters is infeasible on a large Web graph, we use probabilistic counting [Cohen 1997; Flajolet and Martin 1985; Durand and Flajolet 2003]. As to this point, we propose a refinement of the classical probabilistic counting algorithm proposed in [Flajolet and Martin 1985] and adopted in [Palmer et al. 2002]. Our probabilistic counting algorithm turns out to be more accurate than [Palmer et al. 2002] 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 as a generalization of the streaming algorithm used for PageRank computation [Page et al. 1998; Haveliwala 1999]. As a theoretical contribution, the probabilistic analysis of our base algorithm turns out to be considerably more simple than the one given in [Flajolet and Martin 1985] for the original one.

4.1 General algorithm

We start by assigning a random vector of k 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 8, two iterations are shown. On each iteration, a bit set to `1` in any page can only move by one link in distance.

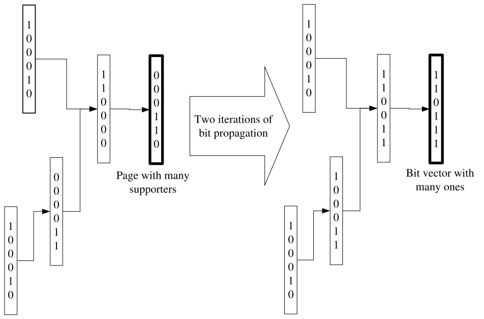

100010
110000
000110
Page with many supporters
000011
100010
Two iterations of bit propagation
100010
110011
110111
Bit vector with many ones
100011
100010

Fig. 8. Schematic depiction of the bit propagation algorithm with two iterations.

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 x has a larger number of distinct supporters than another page y, we expect the bit vector associated to x to contain in the average more `1`s than the bit vector associated to y.

The algorithm, presented in Figure 9, 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 6, allowing the estimation of the number of supporters for all vertices in the graph to be computed concurrently with the execution of Truncated PageRank and PageRank.

The basic algorithm requires $\mathrm{O}(kN)$ 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 $\mathrm{O}(d \log N_{\max}(d))$ times on average, where $N_{\max}(d)$ is the maximum number of supporters at distance at most d. In general, $N_{\max}(d)$ is normally much smaller than N, for the values of d that are useful for 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 $N(x, d) = |S(x, d)|$ and $N_{\max}(d) = \max_x N(x, d)$. 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.

```
Require: $N$: number of nodes, d: distance, k: bits
for node : 1 ... $N$ do {Every node}
    for bit : 1 ... k do {Every bit}
        INIT(node,bit)
    end for
end for
for distance : $1 \ldots$ d do {Iteration step}
    Aux ← $\mathbf{0}_k$
    for src : 1 ... $N$ do {Follow links in the graph}
        for all links from src to dest do
            Aux[dest] ← Aux[dest] OR V[src,·]
        end for
    end for
    for node : 1 ... $N$ do
        V[node,·] ← Aux[node]
    end for
end for
for node: $1 \ldots N$ do {Estimate supporters}
    Supporters[node] ← ESTIMATE( V[node,·] )
end for
return Supporters
```

Fig. 9. **Bit-Propagation Algorithm** for estimating the number of distinct supporters at distance $\leq d$ of all the nodes in the graph simultaneously.

4.2 Base estimation technique

A simple estimator can be obtained as follows:

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).

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 $X_i(x)$ be its i-th component, $i = 1, \ldots, k$. By the properties of the OR operator and by the independence of the $X_i(x)$'s we have,

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

Then, if $B_\epsilon(x) = \sum_{i=1}^{k} X_i(x)$, the following lemma obviously holds:

Lemma 1. *For every x, if every bit of x's label is set to 1 independently with probability ϵ:*

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

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:

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

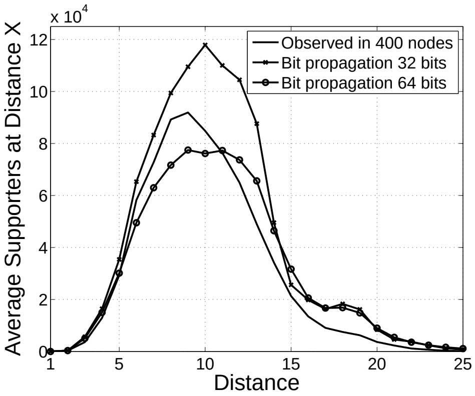


Fig. 10. Comparison of the estimation of the average number of distinct supporters, against the observed value, in a sample of nodes of the `eu-int` graph.

In Figure 10 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 distribution of the *average* number of neighbors. However, this is not good enough for our purposes, as we are interested in specific nodes. This motivates our 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:

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

where the approximation is very good for all values of $N(x, d)$ of practical interest. Also, as a function of ϵ, $\mathbf{E}[B_\epsilon(x)]$ is monotone increasing in the interval $(0, 1]$. This means that, if we consider a decreasing sequence of values of ϵ and the corresponding realizations of $B_\epsilon(x)$, we can reasonably expect to observe more than $(1 - 1/e)k$ bits set to 1 when $\epsilon > 1/N(x)$, with a transition to a value smaller than $(1 - 1/e)k$ when ϵ becomes sufficiently smaller than $1/N(x)$.

In practice, we apply the basic algorithm $O(\log(N))$ times as explained in Figure 11. The basic idea is as follows: starting with a value $\epsilon_{\min}$ (for instance, $\epsilon_{\max} = 1/2$) we proceed by halving ϵ at each iteration, up to some value $\epsilon_{\min}$ (for instance, $\epsilon_{\min} = 1/N$). 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, when ϵ decreases from $2\epsilon(x)$ to $\epsilon(x)$, the value observed for $B_\epsilon(x)$ should transit from a value larger to a value smaller than $(1 - 1/e)k$. This does not hold deterministically of course, but we can prove that it holds with a sufficiently high probability, if k is large enough and $N(x)$ exceeds some suitable constant.

```
Require: $\epsilon_{\min}$, $\epsilon_{\max}$ limits
$\epsilon \leftarrow \epsilon_{\max}$
while $\epsilon > \epsilon_{\min}$ and not all nodes have estimations do
    Run the Bit-Propagation algorithm with $\epsilon$
    for $x$ such that $B_\epsilon(x) < (1 - 1/e)\,k$ for the first time do
        Estimate $\overline{N}(x) \leftarrow 1/\epsilon$
    end for
    $\epsilon \leftarrow \epsilon/2$
end while
return $\overline{N}(x)$
```

Fig. 11. **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.

The following lemma follows immediately:

Lemma 2. *Algorithm* **Adaptive Bit-Propagation** *iterates a number of times that is at most* $\log_2(\epsilon_{\max}/\epsilon_{\min}) \leq \log_2 N$.

The following theorem shows that the probability of $\overline{N}(x)$ deviating from $N(x)$ by more than a constant decades exponentially with k. The proof requires some calculus, but the rough idea is rather simple: considering any page x, the probability that $B_\epsilon(x)$ becomes smaller than $(1-1/e)k$ for any value $\epsilon > c/N(x)$, $c > 1$ a suitable constant, decades exponentially with k. Conversely, the probability that the observed value of $B_\epsilon(x)$ never becomes smaller than $(1-1/e)k$ before ϵ reaches a value smaller than $1/bN(x)$, $b > 1$ being a suitable constant, is again exponentially decreasing in k. For the sake of readibility, the proof of the theorem has been moved to the appendix.

Theorem 1.

$$\mathbf{P}\left[(\overline{N}(x) > 3N(x)) \bigcup \left(\overline{N}(x) < \frac{N(x)}{3}\right)\right] \leq \log_2 N(x) e^{-0.027k} + e^{-0.012k},$$

for every page x such that $N(x) \geq 10$[1].

In the sequel, we denote by $F_<(x)$ the first value of ϵ such that $B_\epsilon(x) < (1-1/e)k$, i.e. $B_\epsilon(x) \geq (1-1/e)k$ for $\epsilon = 2F_<(x), 4F_<(x), \ldots, \epsilon_{\max}$. The following theorem bounds the expected number of times the algorithm reads the graph:

Theorem 2. *If* $0.0392\,k \geq \ln N + \ln \log_2 N$, *the Adaptive Bit-Propagation algorithm reads the graph* $\mathrm{O}(d \log N_{\max}(d))$ *times in the average in order to compute the number of supporters within distance d.*

[1] For $N(x) < 10$ the bound is still exponentially decreasing in k, but the constants in the exponent are lower and we cannot guarantee high accuracy for typical values of k.

PROOF. It is enough to prove that the **while** cycle of the **Adaptive Bit-Propagation** algorithm is iterated $O(\log N_{\max}(d))$ times in the average when estimating the number of supporters withing distance d. Let $I(d)$ be the number of iterations and note that $I(d) \leq \log_2 N$ deterministically by Lemma 2. We start by proving that, for every page x, $F_<(x) \geq \epsilon(x)/4$ with high probability. To this aim, observe that

$$\mathbf{E}\big[B_{\epsilon(x)/4}(x)\big] = k - k\left(1 - \frac{\epsilon(x)}{4}\right)^{N(x)} \leq k - k\left(1 - \frac{1}{4N(x)}\right)^{N(x)} \leq 0.25\,k$$
$$< \frac{1}{2}\left(1 - \frac{1}{e}\right)k,$$

where the third inequality follows recalling Fact 1 and computing the expression for $N(x) = 1$. As a consequence:

$$\mathbf{P}\left[B_{\epsilon(x)/4}(x) > \left(1 - \frac{1}{e}\right)k\right] \leq \mathbf{P}\big[B_{\epsilon(x)/4}(x) \geq 2\mathbf{E}\big[B_{\epsilon(x)/4}(x)\big]\big] \leq e^{-\frac{\mathbf{E}\left[B_{\epsilon(x)/4}(x)\right]}{3}}$$
$$\leq e^{-0.0392\,k},$$

where the second inequality follows by the Chernoff bound with $\delta = 1$, while the third follows since

$$\mathbf{E}\big[B_{\epsilon(x)/4}(x)\big] = k - k\left(1 - \frac{\epsilon(x)}{4}\right)^{N(x)} \geq k - k\left(1 - \frac{1}{8N(x)}\right)^{N(x)} > 0.0392\,k.$$

Here, the second inequality follows since $\epsilon(x) \geq 1/2N(x)$ by definition while, using Fact 1, $(1 - 1/8N(x))^{N(x)}$ is upper bounded by letting $N(x) \to \infty$. As a consequence,

$$\mathbf{P}[\exists x :\, F_<(x) < \epsilon(x)/4] \leq \mathbf{P}\left[\bigcup_{j=1}^{N} F_<(j) < \epsilon(x)/4\right] \leq Ne^{-0.0392\,k} < \frac{1}{\log_2 N}$$

whenever $0.0392\,k \geq \ln N + \ln\log_2 N$. Hence we have:

$$\mathbf{E}[I(d)] \leq \log_2 N\mathbf{P}[\exists x :\, F_<(x) < \epsilon(x)/4] + \max_x \log_2\left(\frac{4\epsilon_{max}}{\epsilon(x)}\right)$$
$$\leq 4 + \log_2(N_{\max}(d)).$$

□

Discussion. Other solutions can be used to compute the number of supporters. One is adopting streaming techniques to compute frequency moments. After the seminal paper by Flajolet and Martin [Flajolet and Martin 1985] extensions and refinements have been proposed, in particular the systematic effort of [Alon et al. 1999]. Differently from these contributions, we do not need to store or compute the values of hash functions or generate node labels following complicated distributions, e.g. exponential ones, as in [Flajolet and Martin 1985]. This also implies that the amount of information kept per node is extremely small and only elementary operations (i.e. bit-wise `OR`s) need to be performed. The initialization step of every phase requires to generate random labels for the nodes. Here, the built-in

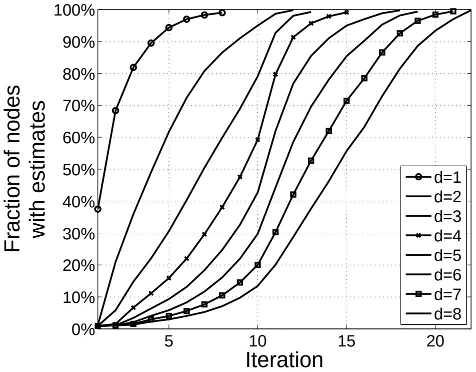


Fig. 12. Fraction of nodes with good estimations after a certain number of iterations. For instance, when measuring at distance $d = 4$, 15 iterations suffice to have good estimators for 99% of the pages (UK-2002 sample).

random number generator is perfectly suited to the purpose, as also experimental results suggest. Furthermore, since every bit is set to 1 with the same probability ϵ, independently of other bit positions and vertices, when ϵ is small this process, as discussed earlier in this section, can be implemented efficiently using a geometric distribution. Other potentially interesting approaches, such as Bloom filters [Broder and Mitzenmacher 2003], are not suited to our purposes, since they require an excessive amount of memory.

The analysis of our algorithms turns out to be much more simple than the ones presented in [Flajolet and Martin 1985] and we hope it provides a more intuitive explanation of why probabilistic counting works. The reason is that the probabilistic analysis in [Flajolet and Martin 1985] requires the average position of the least significant bit that is not set to 1 to be computed 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.

4.4 Experimental results of the bit propagation algorithms

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 $B_\epsilon(x) < (1-1/e)k$, and stop the iterations when 1% or less nodes have $B_\epsilon(x) \geq (1-1/e)k$.

Figure 12 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. The results in this figure are obtained in the UK-2002 collection with 18.5 million nodes (see Section 5).

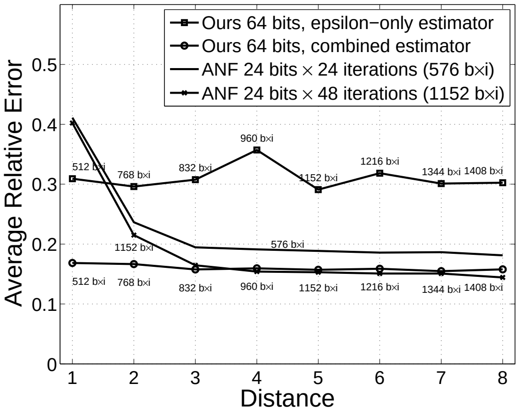


Fig. 13. Comparison of the average relative error of the different strategies (`UK-2002 sample`).

Besides the estimator described in the previous section, we considered the following one: whenever $B_\epsilon(x) < (1 - 1/e)k$ 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 [Palmer et al. 2002]. 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 power of 2 for the 18.5 million nodes of `UK-2002` (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 requirement as our algorithm at distance 5 (one plus the maximum distance we use for spam detection in the experiments shown in section 5).

The comparison is shown in Figure 13. 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 to be more efficient, since it has the same performance but it uses a smaller overall amount of bits.

It is also important to point out 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 with the large data sets we are considering.

5. EXPERIMENTAL FRAMEWORK

In this section we consider several link-based metrics, whose computation uses algorithms which are feasible for large-scale Web collections, and which we have found useful for the purpose of Web spam classification. These are not all possible statistics that can be computed, for a survey of Web metrics, see [Costa et al. 2005].

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 or not. We use these metrics to build a set of classifiers, that we use to test the fitness of each metric to the purpose of automatic spam classification.

5.1 Data sets

We use two large subsets of pages from the `.uk` domain, downloaded in 2002 and 2006 by the *Dipartimento di Scienze dell'Informazione*, *Università degli studi di Milano*. These collections are publicly available at `http://law.dsi.unimi.it/`, and were obtained using by a breadth-first visit using the UbiCrawler [Boldi et al. 2004].

Table I summarizes the properties of these collections:

Table I. Characteristics of the base collections used in our experiments.

Collection	Pages	Links	Links/page	Hosts	Pages/host
`UK-2002`	18.5 M	298 M	16.1	98,542	187.9
`UK-2006`	77.9 M	3,022 M	38.8	11,403	6828.3

The `UK-2006` collection is much deeper than the `UK-2002` collection, but it includes less hosts, as it was given a smaller set of seeds to start with. The fact that the `UK-2006` collection also has much more links per page agrees with empirical observations that the Web is becoming denser in general [Leskovec et al. 2005].

5.2 Data labelling

Due to the large size of this collection, we decided to classify entire hosts instead of individual pages. This increases coverage of the sample, but it also introduces errors, as there are some hosts that consist of a mixture of spam pages and legitimate contents.

—`UK-2002`: The manual classification was done by one of the authors of this paper and took roughly three days of work. Whenever a link farm was found inside a host, the entire host was marked as spam. Initially we sampled at random but, as the ratio of spam pages to normal pages was small and we wanted to have many spam examples, we actively searched for spam pages in our collection, so this sample is not uniformly random. We classified a group of the hosts with the higher PageRank in their home page, with the higher overall PageRank and with the larger number of pages. Other hosts were added by classifying all the hosts with the larger hostname length, as several spammers tend to create long names such as "`www.buy-a-used-car-today.example`" (but not all sites with long host names were spam). For the same reason, we searched for typical spamming terms in the host names, and we classified all the hosts with domain names including keywords such as `mp3`, `mortgage`, `sex`, `casino`, `buy`, `free`, `cheap`, etc. (not all of them had link farms) and typical non-spam domain such as `.ac.uk`, `.gov.uk` and others.

For the purposes of the classifiers, we take only the "normal" and "spam" labels into consideration. This diverts from the use of this collection in [Becchetti et al. 2006a; 2006b] in which we considered an extra label, "suspicious" as "normal", but is done for consistency with the experiments in the other collection. In any case, "suspicious" labels were used in only 3% of the cases.

—`UK-2006`: The manual classification was done by a group of over 20 volunteers who received a set of standard guidelines, as described in [Castillo et al. 2006]. These guidelines cover most of the types of Web spam mentioned in the literature, not purely link-based spam. The sampling was done uniformly at random over all the hosts in the collection, assigning two judges to each host in most cases. A third judgement was used to break ties in the case of contradictory evaluations (i.e.: one normal and one spam label). The collection also includes automatic marks based in several domains such as `.ac.uk`, `.gov.uk`, `.police.uk`, etc.

For the purposes of the classifier, we did a majority vote among normal and spam judgments (ignoring borderline evaluations). We kept all hosts that matched our domain-based patterns or in which there were at least 2 human judges.

Table II summarizes the number of labels on each collection.

Table II. Characteristics of the labels used on each collection.

	Classified	Normal	Spam
Collection	Hosts	(%)	(%)
`UK-2002`	5,182	4,342 (84%)	840 (16%)
`UK-2006`	5,622	4,948 (78%)	674 (12%)

5.3 Classification caveats

There are many Web sites whose design is optimized for search engines, but which also provide useful content. There is no clear line to divide a spam site from a heavily optimized Web site, designed by a person who knows something about how search engines work.

In the case of `UK-2002`, we examined the current contents of the pages and not the contents of them in 2002 (as those were not available). This can negatively affect the results in the UK-2002 collection and introduce extra noise. Also, in the UK-2002 the classification was based solely on link-based spam, not other forms of spam.

In the case of UK-2006, the evaluation was done less than 2 months after the crawling, and the judges had access to the home page of the page at crawling time, pulled automatically from the crawler's cache. The evaluation guidelines covered most types of spam including both keyword-based and link-based spam.

5.4 Automatic classification

We automatically extracted a series of features from the data, including the PageRank, TruncatedPageRank at distance $d = 1, 2, 3$ and 4, and the estimates 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 samples, the home page of the host is the page with the highest PageRank in 38% of the cases (UK-2002) and 57% of the cases (UK-2006). In the case of hosts marked as spam, the proportions are 77% and 58% respectively.

The labelled 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 [Witten and Frank 1999] implementation of C4.5 decision trees. Describing this classifiers here in detail in not possible due to space limitations, for a description see [Witten and Frank 1999].

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 [Baeza-Yates and Ribeiro-Neto 1999], considering the spam detection task:

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

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

For combining precision (P) and recall (R), we used the **F-Measure**, which corresponds to the harmonic mean of these two numbers,

$$F = \frac{2PR}{P + R}$$

We also measured the two types of errors in spam classification:

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

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

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.

For each set of features we built a classifier, we did not use pruning and let weka generate as many rules as possible as long as there are at least 2 hosts per leaf (this is the M parameter in the `weka.classifiers.trees.J48` implementation).

We also used bagging [Breiman 1996], a technique that creates many classifiers (in our case, 10), and then uses majority voting for deciding the class to which an element belongs. The classifiers that use bagging perform in general better than the individual classifiers they are composed of.

6. EXPERIMENTAL RESULTS

This section presents the experimental results obtained by creating automatic classifiers with different sets of attributes. At the end of this section we present the performance of a classifier that uses the entire set of link-based features.

6.1 Degree-based measures

The distribution of in-degree and out-degree can be obtained very easily by reading the Web graph only once. In Figure 14 we depict the histogram of this metric over the normal pages and the spam pages. The histogram is shown with bars for the normal pages and with lines for the spam pages. Both histograms are normalized independently, and the y-axis represents frequencies.

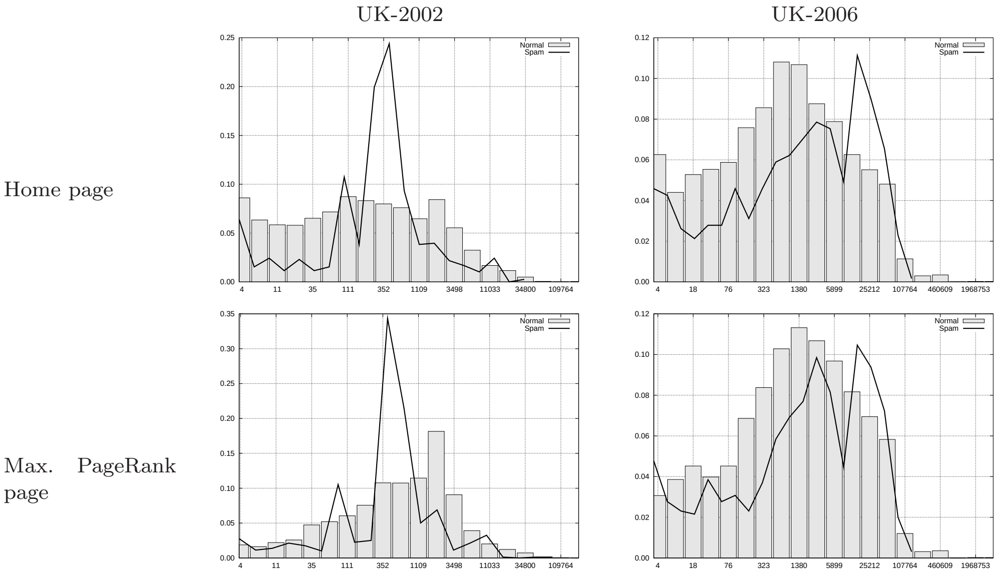


Fig. 14. Histogram of the log(indegree) of home pages.

In the case of spam hosts in the `UK-2002` collection, there is a large group of about 40% of them that have an in-degree in a very narrow interval. In the `UK-2006` the in-degree seems to be higher on average for spam pages, but there is no dramatic "peak" as in `UK-2002`.

Another degree-based metric is the **edge-reciprocity**. This measures how many of the links in the directed Web graph are reciprocal. The edge-reciprocity can be computed easily by simultaneously scanning the graph and its transposed version, and measuring the overlap between the out-neighbors of a page and its in-neighbors. Figure 15 depicts the edge reciprocity in both collections. This metric appears to be take extreme values (0 and 1) with high frequency; this is because the degree of the pages follows a power-law, and there are many pages with degree 1..

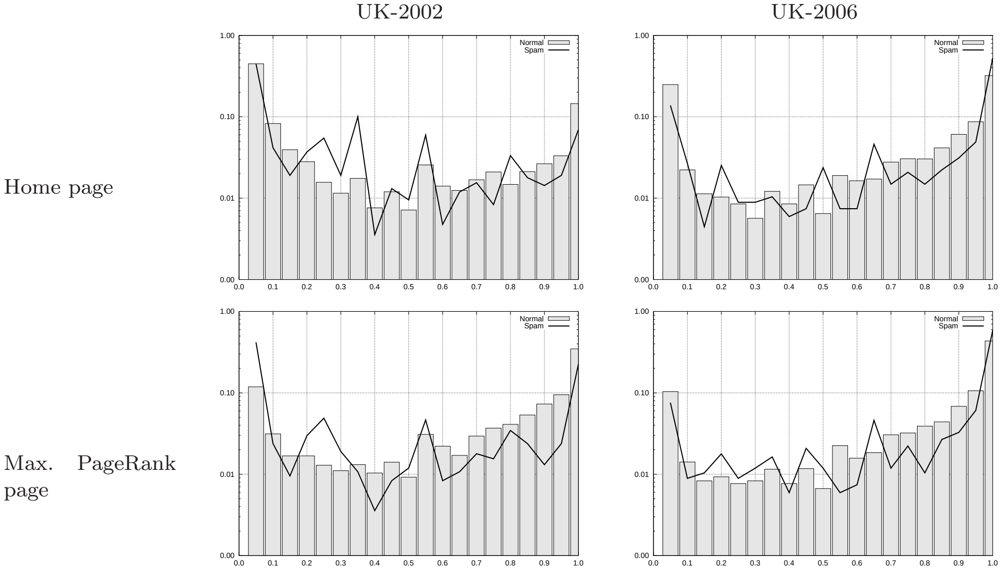


Fig. 15. Histogram of the edge-reciprocity of home pages.

The degree of the nodes induces a natural “hierarchy” that can be used to define different classes of nodes. A network in which most nodes are connected to other nodes in the same class (for instance, most of the connections of highly-linked are to other highly-linked nodes) is called “**assortative**” and a network in which the contrary occurs is called “**disassortative**”. This distinction is important from the point of view of epidemics [Gupta et al. 1989].

We measured for every host in our sample the ratio between its degree and the average degree of its neighbors (considering both in- and out-links). In Figure 16 we can see that in both collections there is a mixing of assortative and disassortative behavior. The home pages of the spam hosts tend to be linked to/by pages with relatively lower in-degree. This is clearer in the `UK-2002` sample where there is a peak at 10, meaning that for that group, their degree is 10 times larger than the degree of their direct neighbors.

We used the following attributes in the home page and the page with maximum PageRank, plus a binary variable indicating if they are the same page ($8 \times 2 + 1 = 17$ features in total):

(1) Log of the indegree

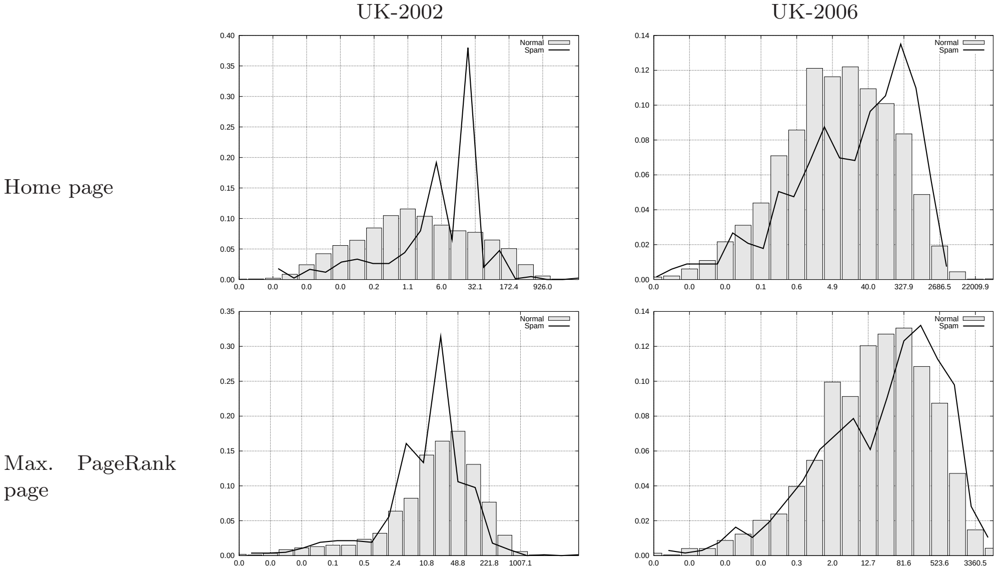


Fig. 16. Histogram of the degree/degree ratio of home pages.

(2) Log of the outdegree
(3) Reciprocity
(4) Log of the assortativity coefficient
(5) Log of the average in-degree of out-neighbors
(6) Log of the average out-degree of in-neighbors
(7) Log of the sum of the in-degree of out-neighbors
(8) Log of the sum of the out-degree of in-neighbors

On Table III we report on the performance of a C4.5 decision tree with bagging, using only degree-based features. The performance is acceptable in the UK-2002 dataset but very poor in the UK-2006 dataset. This means that in the UK-2002 dataset there are many spam hosts that have anomalous local connectivity, while these hosts are fewer in the UK-2006 data.

Table III. Performance using only degree-based attributes

Dataset	True positives	False positives	F-Measure
UK-2002	0.732	0.015	0.808
UK-2006	0.323	0.024	0.432

6.2 PageRank

We calculated the PageRank scores for the pages in the collection using $\alpha = 0.85$ and the formula at the beginning of Section 3. We plot the distribution of the PageRank values of the home pages in Figure 17. We can see a large fraction of pages sharing the same PageRank. This is more or less expected, as there is also a large fraction of pages sharing the same in-degree (although these are not equivalent metrics).

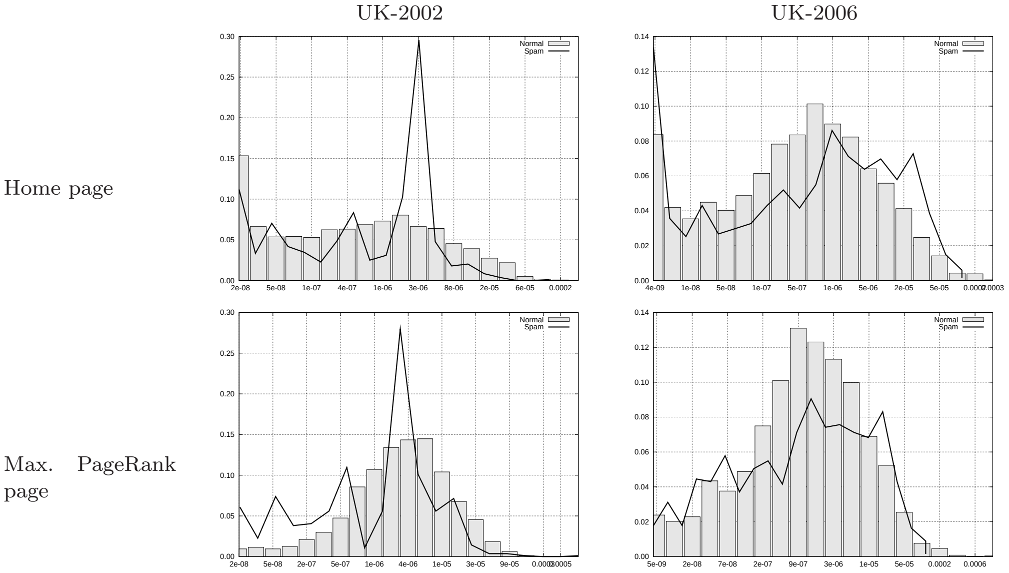


Fig. 17. Histogram of the PageRank values.

Following an idea by Benczúr et al. [Benczúr et al. 2005], we studied the PageRank distribution of the pages that contribute to the PageRank of a given page. In [Benczúr et al. 2005], this distribution is studied over a sample of the pages that point recursively to the target page, with a strong preference for shorter paths.

We calculate the standard deviation of the PageRank values of the in-neighbors of pages. The result is shown in Figure 18, and it seems that for a large group of spammers in our datasets, it is more frequent to have less dispersion in the values of the PageRank of the in-neighbors than in the case of non-spam hosts.

We used the degree-based attributes from the previous section, plus the following measured in the home page and the page with maximum PageRank, plus the PageRank of the home page divided by the PageRank of the page with the maximum PageRank. This makes a total of 28 features ($17 + 5 \times 2 + 1 = 28$):

(1) Log of PageRank
(2) Log of (in-degree divided by PageRank)
(3) Log of (out-degree divided by PageRank)
(4) Standard deviation of PageRank of in-neighbors
(5) Log of (standard deviation of PageRank of in-neighbors divided by PageRank)

The performance of an automatic classifier using degree- and PageRank-based attributes is reported in Table IV. In both collections the performance improves by adding these features.

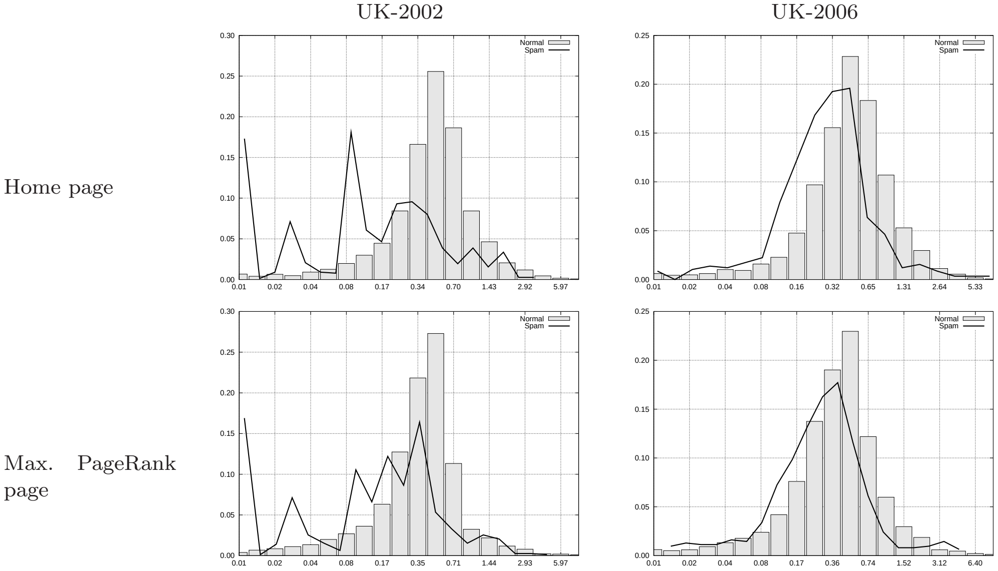


Fig. 18. Histogram of the standard deviation of the PageRank of neighbors.

Table IV. Performance using only degree-based and Pagerank-based attributes

Dataset	True positives	False positives	F-Measure	Previous F-Measure from Table III
UK-2002	0.768	0.014	0.835	0.808
UK-2006	0.359	0.025	0.466	0.432

6.3 TrustRank

In [Gyöngyi et al. 2004] the TrustRank algorithm for trust propagation is described: it starts with a seed of hand-picked **trusted** nodes and then propagates their scores by simulating a random walk with restart to the trusted nodes. The intuition behind TrustRank is that a page with high PageRank, but lacking a relationship with any of the trusted pages, is suspicious.

The **spam mass** of a page is defined as the amount of PageRank received by that page from spammers. This quantity cannot be calculated in practice, but it can be estimated by measuring the **estimated non-spam mass**, which is the amount of score that a page receives from trusted pages. For the purpose of this paper we refer to this quantity simply as the **TrustRank score** of a page.

For calculating this score, a biased random walk is carried out on the Web graph. With probability α we follow an out-link from a page, and with probability $1 - \alpha$ we go back to one of the trusted nodes picked at random. For the trusted nodes we used data from the Open Directory Project (available at `http://rdf.dmoz.org/`), selecting all the listed hosts belonging to the `.uk` domain. This includes over 150,000 different hosts, from which we removed the hosts that we know were spam (21 hosts in `UK-2002` and 29 hosts in `UK-2006`).

For the UK-2002 sample 32,866 ODP hosts were included in our collection, this is 33% of the known hosts in our collection. We used the same proportion (33% of known hosts) for UK-2006, sampling 3,800 ODP hosts in this case.

As shown in Figure 19, the score obtained by the home page of hosts in the normal class and hosts in the spam class is very different. Also, the ratio between the TrustRank score and the PageRank (the estimated relative non-spam mass, shown in Figure 20) is also very effective for separating spam from normal pages.

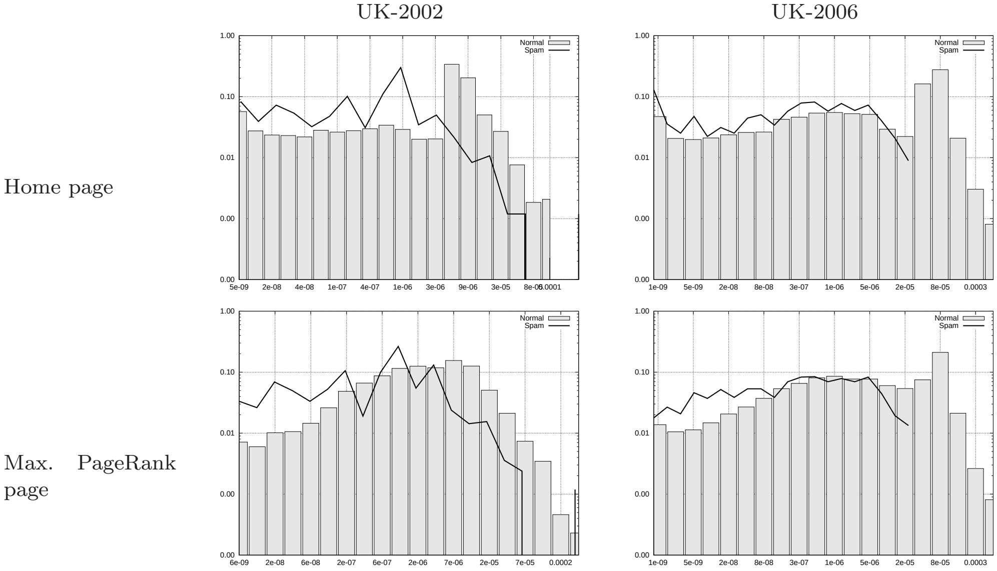


Fig. 19. Histogram of TrustRank scores (absolute).

We build a classifier using the attributes from the previous section, plus the following attributes measured in the home page and the page with the maximum PageRank, plus the TrustRank of the home page divided by the TrustRank of the page with the maximum PageRank ($28 + 3 \times 2 + 1 = 35$ attributes):

(1) Log of TrustRank (log of absolute non-spam mass)

(2) Log of (TrustRank divided by PageRank) (log of relative non-spam mass)

(3) Log of (TrustRank divided by in-degree)

The performance of an automatic classifier using metrics based on degree, PageRank, and TrustRank, is shown in Table V. The performance improvement is noticeable in the UK-2006 collection.

Table V. Performance using attributes based on degree, PageRank and TrustRank

Dataset	True positives	False positives	F-Measure	Previous F-Measure from Table IV
UK-2002	0.786	0.014	0.846	0.835
UK-2006	0.539	0.037	0.595	0.466

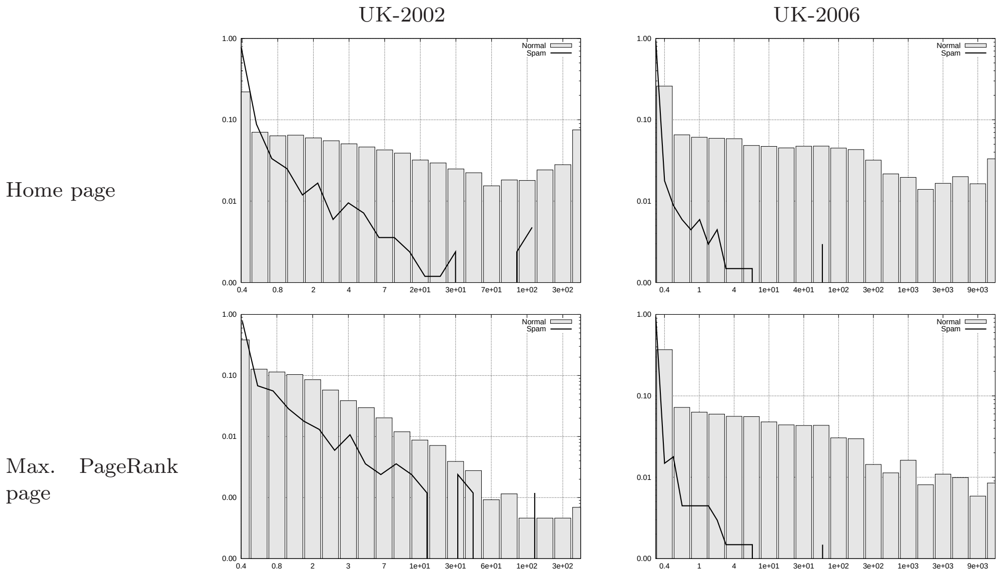


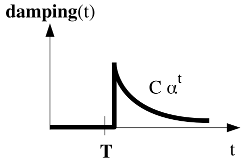
Fig. 20. Histogram of TrustRank scores (relative to PageRank).

6.4 Truncated PageRank

In [Becchetti et al. 2006b] we described Truncated PageRank, a link-based ranking function that reduces the importance of neighbors that are considered to be topologically "close" to the target node. In [Zhang et al. 2004] 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.

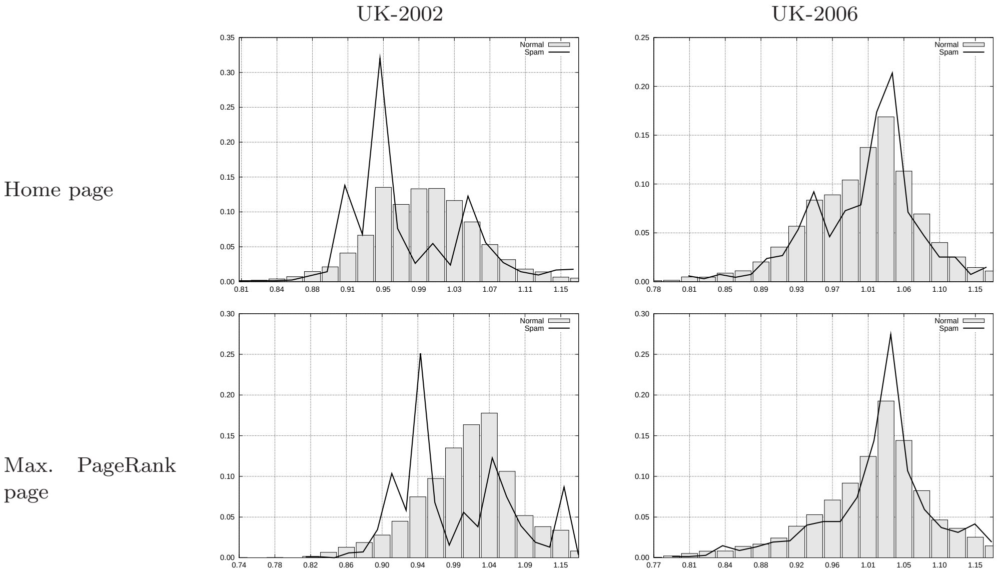
Intuitively, a way of demoting spam pages is to consider a damping function that **removes the direct contribution of the first levels of links**, such as:

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



Where C is a normalization constant and α is the damping factor used for PageRank. 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. The calculation of Truncated PageRank is described in detail in [Becchetti et al. 2006b]. There is a very fast method for calculating Truncated PageRank. Given a PageRank computation, we can store "snapshots" of the PageRank values at different iterations and then take the difference and normalize those values at the end of the PageRank computation. Essentially, this means that the Truncated PageRank can be calculated for free during the PageRank iterations.

Note that as the number of indirect neighbors also depends on the number of direct neighbors, reducing the contribution of the first level of links by this method does not mean that we are calculating something completely different from PageRank. In fact, for most pages, both measures are closely correlated, as shown in [Becchetti et al. 2006b].


UK-2002

UK-2006

Home page

Max. PageRank page

Fig. 21. Histogram of maximum change in TruncatedPageRank up to four levels.

In practice, we observe that for the spam hosts in the UK-2002 collection, the Truncated PageRank is smaller than the PageRank. If we observe the ratio of Truncated PageRank at distance i versus Truncated PageRank at distance $i - 1$, as shown in Figure 21, we can see a difference between the spam and non-spam classes, but this difference is not present in the UK-2006 collection.

We built a classifier using the degree- and PageRank- based attributes, plus the following in the home page and the page with the maximum PageRank:

(1) Log of Truncated PageRank at distance 1, 2, 3, and 4 (4 features)

(2) Log of: Truncated PageRank at distance T, divided by Truncated PageRank at distance $T - 1$, for $T = 2, 3, 4$ (3 features)

(3) Log of: Truncated PageRank at distance T, divided by PageRank, for $T = 1, 2, 3, 4$ (4 features)

(4) Log of the minimum, average, and maximum change of: Truncated PageRank T divided by Truncated PageRank $T - 1$, for $T = 1, 2, 3, 4$, considering that Truncated PageRank at distance 0 is equal to PageRank (3 features)

Additionally we used the Truncated PageRank at distance T at the home page, divided by Truncated PageRank at distance T in the page with maximum PageRank, for $T = 1, 2, 3, 4$. The total number of features of this classifier is $28 + (4 + 3 + 4 + 3) \times 2 + 4 = 60$.

Table VI. Performance using attributes based on degree, PageRank and Truncated PageRank

Dataset	True positives	False positives	F-Measure	Previous F-Measure from Table IV
UK-2002	0.783	0.015	0.843	0.835
UK-2006	0.355	0.020	0.473	0.466

The performance obtained with this classifier is shown in Table VI. Its improvement over the classifier based in degree- and PageRank-based metrics, is lower than the one obtained by using TrustRank.

6.5 Estimation of supporters

In this section we use the technique for estimating supporters presented in Section 4. This algorithm can be very easily expanded upon to consider the number of different **hosts** contributing to the ranking of a given host. To do so, in the initialization the bit masks of all the pages in the same host are made equal.

We found that the estimation of supporters hosts is very valuable for separating spam from non-spam, in particular when the *rate of change* of the number of supporters is studied. Figure 22 shows the minimum, and Figure 23 the maximum of this quantity for the counting of different hosts.

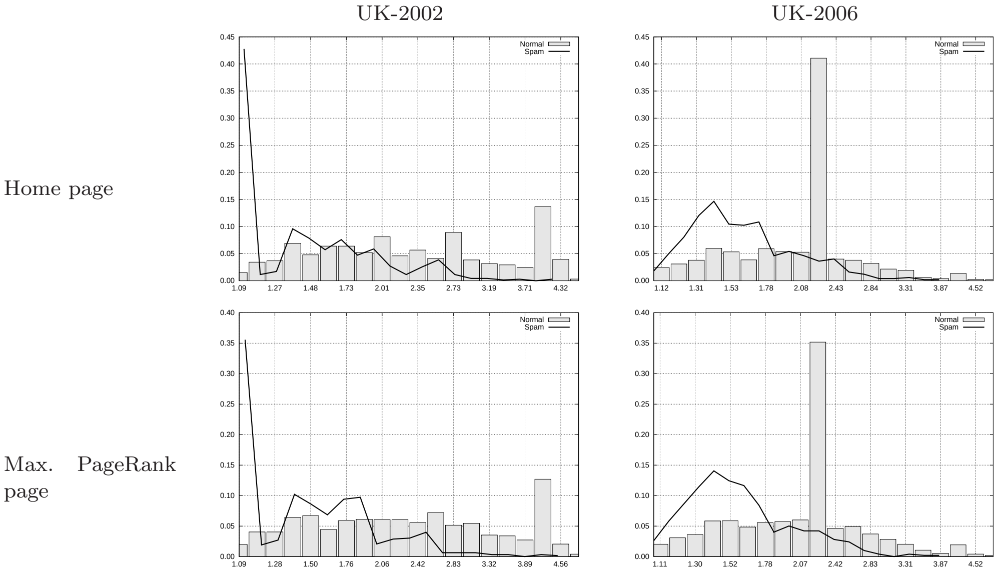


Fig. 22. Histogram of minimum change of site neighbors.

We built a classifier using the degree- and PageRank-based attributes, plus the following attributes in the home page and the page with the maximum PageRank:

(1) Log of the number of supporters (different hosts) at distance $d = 1, 2, 3, 4$ (4 features)

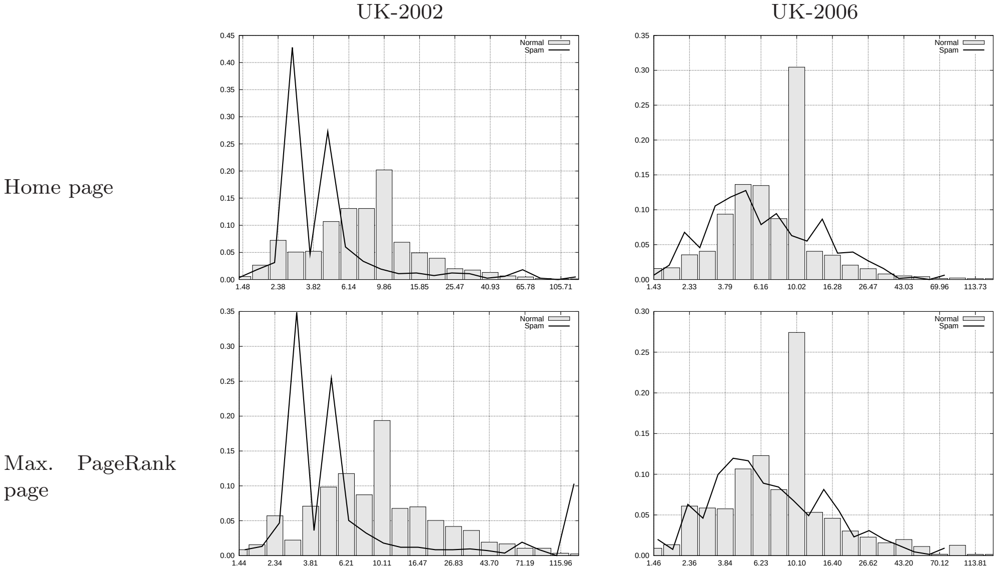


Fig. 23. Histogram of maximum change of site neighbors.

(2) Log of: the number of supporters (different hosts) at distance $d = 1, 2, 3, 4$ divided by PageRank (4 features)

(3) Log of: the number of supporters (different hosts) at distance $d = 2, 3, 4$ divided by number of supporters (different hosts) at distance $d - 1$ (3 features)

(4) Log of the minimum, maximum, and average of: the number of supporters (different hosts) at distance $d = 2, 3, 4$, divided by number of supporters (different hosts) at distance $d - 1$ (3 features)

(5) Log of: the number of supporters (different hosts) at distance exactly $d = 2, 3, 4$ (that is, the number of supporters at distance d minus the number of supporters at distance $d - 1$), divided by PageRank (3 features)

(6) Log of the number of supporters at distance $d = 2, 3, 4$; note that supporters at distance 1 is in-degree (3 features)

(7) Log of: the number of supporters at distance $d = 2, 3, 4$ divided by PageRank (3 features)

(8) Log of: the number of supporters at distance $d = 2, 3, 4$ divided by number of supporters at distance $d - 1$ (3 features)

(9) Log of the minimum, maximum, and average of: the number of supporters at distance $d = 2, 3, 4$, divided by number of supporters at distance $d - 1$ (3 features)

(10) Log of: the number of supporters at distance exactly $d = 2, 3, 4$, divided by PageRank (3 features)

Additionally we included the ratio of the number of supporters (different hosts) in the home page and the page with the maximum PageRank, at distance $d = 1, 2, 3, 4$, and the same ratio for the number of supporters at distance $d = 2, 3, 4$. The total is $28 + 32 \times 2 + 4 + 3 = 99$ features.

The performance of the classifier that uses these attributes is shown in Table VII.

Table VII. Performance using attributes based on degree, PageRank and Estimation of Supporters

Dataset	True positives	False positives	F-Measure	Previous F-Measure from Table IV
UK-2002	0.802	0.009	0.867	0.835
Only pages	0.796	0.013	0.854	
Only hosts	0.779	0.010	0.850	
UK-2006	0.466	0.032	0.548	0.466
Only pages	0.401	0.029	0.496	
Only hosts	0.467	0.029	0.556	

In the table, we have also included the performance of the classifier by reducing the number of attributes to count only different hosts, or only different pages. In the UK-2002 collection, it is better to count supporters directly, while in the UK-2006 collection, it is better to use host-based counts instead.

6.6 Combined classifier

By combining all of the attributes we have discussed so far (163 attributes in total), we obtained a better performance than we did for each of the individual classifiers. Table VIII presents the results of the combined classifier along with the results obtained with the previous classifiers.

Table VIII. Summary of the performance of the different classifiers studied on this paper

		UK-2002		UK-2006	
Section	Metrics	True positives	False positives	True positives	False positives
6.1	Degree (D)	0.732	0.015	0.323	0.024
6.2	D + PageRank (P)	0.768	0.014	0.359	0.025
6.3	D + P + TrustRank	0.786	0.014	0.539	0.037
3	D + P + Trunc. PageRank	0.783	0.015	0.355	0.020
6.5	D + P + Est. Supporters	0.802	0.009	0.466	0.032
6.6	All attributes	0.805	0.009	0.585	0.037

The classifier described on Section 6.2, that uses only degree-based and PageRank-based attributes, can be considered as a baseline. In this case, the best improvements in the UK-2002 collection are obtained using the estimation of supporters, followed by TrustRank, followed by Truncated PageRank; in the UK-2006 collection, the best is TrustRank, followed by estimation of supporters, followed by Truncated PageRank.

7. RELATED WORK

Characterizing and detecting spam: In [Fetterly et al. 2004] 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. In [Gomes et al. 2005] a comparison of link-based statistical properties of spam and legitimate e-mail messages is presented.

The method of "shingles" for detecting dense sub-graphs [Gibson et al. 2005] can be applied for link farm detection, as members of a link farm might share a substantial fraction of their out-links (however, the algorithm will perform worse if the link farm is randomized).

In [Zhang et al. 2004] it is shown that spam pages should be very sensitive to changes in the damping factor of the PageRank calculation; with the case of Truncated PageRank we not only modify the damping factor, but also the whole damping function.

Nepotistic links, that is, links that are present for reasons different than merit, can be detected and removed from Web graphs before applying link-based ranking techniques. This is the approach proposed in [Davison 2000a] and extended in [da Costa-Carvalho et al. 2006]. Another idea is to use "bursts" of linking activity as a suspicious signal [Shen et al. 2006].

In [Benczúr et al. 2005] 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 [Ntoulas et al. 2006; Drost and Scheffer 2005; Davison 2000a] 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. The performance of content-based classification is comparable to our approach.

A content-based classifier described in [Ntoulas et al. 2006], without bagging nor boosting, reported 82% of recall, with 2.5% of false positives (84.4% and 1.3% with bagging, 86.2% and 1.3% with boosting). Unfortunately, their classifier is not publicly available for evaluation on the same collection as ours. Also, note that link-based and content-based approaches to spam detection are orthogonal and suitable for detection of different kinds of spam activity. 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.

Outside the topic of Web spam, links have be used for classification tasks. For instance, [Lu and Getoor 2003] uses the categories of the objects linked from a target page to infer the category of such page.

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 [Wu and Davison 2005] is based on finding a page which is part of a link farm and then marking all pages that have links towards it, possibly recursively following back-links up to a certain threshold (this is also called "BadRank").

In [Benczúr et al. 2005], "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 [Morris 1978] 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 [Flajolet and Martin 1985]; this technique is applied to the particular case of counting the number of in-neighbors or "supporters" of a page in [Palmer et al. 2002]. The use of probabilistic counting is important in this case, as the cost of calculating the exact values is prohibitive [Lipton and Naughton 1989].

8. CONCLUSIONS AND FUTURE WORK

On document classification tasks, the most direct approach is to build automatic classification systems based on the contents and/or formatting of the documents. With regard to the particular task of Web spam classification, we can take a different approach and build automatic classification systems based on their link structure. This is what makes the approach to Web spam we have described in this paper unique. Also, we have been careful to restrict ourselves to attributes that can be obtained from a Web graph using streaming algorithms, so they can be applied to Web graphs of any size.

The performance of our detection algorithms is higher in the UK-2002 collection than in the UK-2006 collection. The latter was labeled with a broader definition of spam that includes also content-based spam in addition to link-based spam. However, we are not suggesting to use only link-based attributes. The link-analysis methods are orthogonal to content-based analysis, and the performance of a classifier using content- and link-based features is substantially better than the performance of a classifier using only one set of features [Castillo et al. 2006].

As a general criticism of our work, our host-based approach has some drawbacks that should be addressed in future work. For instance, hosts can have mixed spam/legitimate content, and it is important to study how frequently this occurs, as well as testing how link-based attributes can help in the classification task at a page level. Also, a better definition of Web site instead of host would be useful; for instance, considering multi-site hosts such as geocities.com as separated entities.

Finally, the use of regularization methods that exploit the topology of the graph and the locality hypothesis [Davison 2000b] is promising, as it has been shown that those methods are useful for general Web classification tasks [Zhang et al. 2006; Angelova and Weikum 2006; Qi and Davison 2006] and that can be used to improve the accuracy of Web spam detection systems [Castillo et al. 2006].

Acknowledgments

We thank Paolo Boldi, Massimo Santini and Sebastiano Vigna for obtaining the Web collections that we use for our work. We also thank Karen Whitehouse for a thorough revision of the English.

REFERENCES

Alon, N., Matias, Y., and Szegedy, M. 1999. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci. 58,* 1, 137–147.

Angelova, R. and Weikum, G. 2006. Graph-based text classification: learn from your neighbors. *Proceedings of the international ACM SIGIR conference*, 485–492.

Baeza-Yates, R., Boldi, P., and Castillo, C. 2006. Generalizing pagerank: Damping functions for link-based ranking algorithms. In *Proceedings of ACM SIGIR*. ACM Press, Seattle, Washington, USA, 308–315.

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

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

Becchetti, L., Castillo, C., Donato, D., Leonardi, S., and Baeza-Yates, R. 2006a. Link-based characterization and detection of Web Spam. In *Second International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*. Seattle, USA.

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

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

Boldi, P., Codenotti, B., Santini, M., and Vigna, S. 2004. Ubicrawler: a scalable fully distributed web crawler. *Software, Practice and Experience 34,* 8, 711–726.

Breiman, L. 1996. Bagging predictors. *Machine Learning 24,* 2, 123–140.

Broder, A. and Mitzenmacher, M. 2003. Network Applications of Bloom Filters: A Survey. *Internet Mathematics 1,* 4, 485–509.

Castillo, C., Donato, D., Becchetti, L., Boldi, P., Leonardi, S., Santini, M., and Vigna, S. 2006. A reference collection for web spam. *SIGIR Forum 40,* 2 (December), 11–24.

Castillo, C., Donato, D., Gionis, A., Murdock, V., and Silvestri, F. 2006. Know your neighbors: Web spam detection using the web topology. Tech. rep.

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

Costa, L., Rodrigues, F. A., Travieso, G., and Villas. 2005. Characterization of complex networks: A survey of measurements.

da Costa-Carvalho, A. L., Chirita, P.-A., de Moura, E. S., Calado, P., and Nejdl, W. 2006. Site level noise removal for search engines. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*. ACM Press, New York, NY, USA, 73–82.

Davison, B. D. 2000a. Recognizing nepotistic links on the Web. In *Artificial Intelligence for Web Search*. AAAI Press, Austin, Texas, USA, 23–28.

Davison, B. D. 2000b. Topical locality in the web. In *Proceedings of the 23rd annual international ACM SIGIR conference on research and development in information retrieval*. ACM Press, Athens, Greece, 272–279.

Demetrescu, C., Finocchi, I., and Ribichini, A. 2006. Trading off space for passes in graph streaming problems. In *Proceedings of the 7th annual ACM-SIAM Symposium on Discrete Algorithms*.

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

Durand, M. and Flajolet, P. 2003. Loglog counting of large cardinalities (extended abstract). In *Proceedings of ESA 2003, 11th Annual European Symposium on Algorithms*. Lecture Notes in Computer Science, vol. 2832. Springer, 605–617.

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

Feigenbaum, J., Kannan, S., Gregor, M. A., Suri, S., and Zhang, J. 2004. On graph problems in a semi-streaming model. In *31st International Colloquium on Automata, Languages and Programming*.

Fetterly, D., Manasse, M., and Najork, M. 2004. 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)*. Paris, France, 1–6.

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

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

Gomes, L. H., Almeida, R. B., Bettencourt, L. M. A., Almeida, V., and Almeida, J. M. 2005. Comparative graph theoretical characterization of networks of spam and legitimate email.

Gori, M. and Witten, I. 2005. The bubble of web visibility. *Commun. ACM 48,* 3 (March), 115–117.

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

Gupta, S., Anderson, R. M., and May, R. M. 1989. Networks of sexual contacts: implications for the pattern of spread of hiv. *AIDS 3,* 12 (December), 807–817.

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

Gyöngyi, Z., Garcia-Molina, H., and Pedersen, J. 2004. Combating Web spam with TrustRank. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB)*. Morgan Kaufmann, Toronto, Canada, 576–587.

Haveliwala, T. 1999. Efficient computation of pagerank. Tech. rep., Stanford University.

Henzinger, M. R., Raghavan, P., and Rajagopalan, S. 1999. Computing on data streams. *Dimacs Series In Discrete Mathematics And Theoretical Computer Science*, 107–118.

Leskovec, J., Kleinberg, J., and Faloutsos, C. 2005. Graphs over time: densification laws, shrinking diameters and possible explanations. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM Press, New York, NY, USA, 177–187.

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

Lu, Q. and Getoor, L. 2003. Link-based classification. In *International Conference on Machine Learning*.

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

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

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

Page, L., Brin, S., Motwani, R., and Winograd, T. 1998. The PageRank citation ranking: bringing order to the Web. Tech. rep., Stanford Digital Library Technologies Project.

Palmer, C. R., Gibbons, P. B., and Faloutsos, C. 2002. 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*. ACM Press, New York, NY, USA, 81–90.

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

Qi, X. and Davison, B. D. 2006. Knowing a web page by the company it keeps. In *Proceedings of the 15th ACM Conference on Information and Knowledge Management (CIKM)*. Arlington, VA, USA, 228–237.

Shen, G., Gao, B., Liu, T.-Y., Feng, G., Song, S., and Li, H. 2006. Detecting link spam using temporal information. In *ICDM*. Hong Kong.

Vitter, J. S. 2001. External memory algorithms and data structures. *ACM Computing Surveys 33,* 2, 209–271.

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

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

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

Zhang, T., Popescul, A., and Dom, B. 2006. Linear prediction models with graph regularization for web-page categorization. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM Press, New York, NY, USA, 821–826.

A. PROOF OF THEOREM 1

In the proof of the theorem we will repeatedly use the following facts:

Fact 1. *For every $\beta > 0$, the function*

$$f(s) = \left(1 - \frac{1}{\beta s}\right)^s$$

is monotonically increasing in the interval $[1/\beta, \infty)$.

Proof. The function (and its derivative) is 0 in $s = 1/\beta$. Also, the function and its derivative are positive in $(1/\beta, \infty)$. □

Fact 2. *For every $n \geq 1$:*

$$\left(1 - \frac{1}{n+1}\right)^{n+1} < \frac{1}{e} < \left(1 - \frac{1}{n+1}\right)^n$$

Proof. This is an easy consequence of the well known fact that

$$\left(1 + \frac{1}{n}\right)^n < e < \left(1 + \frac{1}{n}\right)^{n+1}.$$

□

In the sequel, we denote by $F_<(x)$ the first value of ϵ such that $B_\epsilon(x) < (1-1/e)k$, i.e. $B_\epsilon(x) \geq (1-1/e)k$ for $\epsilon = 2F_<(x), 4F_<(x), \ldots, \epsilon_{\max}$.

Theorem 1

$$\mathbf{P}\left[(\overline{N}(x) > 3N(x)) \bigcup \left(\overline{N}(x) < \frac{N(x)}{3}\right)\right] \leq \log_2 N(x) e^{-0.027k} + e^{-0.012k},$$

for every page x such that $N(x) \geq 10^2$.

Proof. We first consider $\mathbf{P}\left[\overline{N}(x) < (1/3)N(x)\right]$. Note that $\overline{N}(x) = 1/F_<(x)$ by definition of $F_<(x)$. Also, $\overline{N}(x) < (1/3)N(x)$ implies $F_<(x) > 3/N(x) \geq 3\epsilon(x)$

[2] For $N(x) < 10$ the bound is still exponentially decreasing in k, but the constants in the exponent are lower and we cannot guarantee high accuracy for typical values of k.

and this is equivalent to $F_<(x) \geq 4\epsilon(x)$, by definition of $\epsilon(x)$ and by the algorithm. Hence,

$$\mathbf{P}\left[\overline{N}(x) < (1/3)N(x)\right] \leq \mathbf{P}[F_<(x) \geq 4\epsilon(x)] = \sum_{i=2}^{i_{\max}} \mathbf{P}\left[F_<(x) = 2^i\epsilon(x)\right],$$

where $i_{\max} = \log_2(\epsilon_{\max}/\epsilon(x)) \leq \log_2 N(x)$, since $\epsilon(x) \geq 2/N(x)$ by definition. We continue with:

$$\begin{aligned}
&\sum_{i=2}^{i_{\max}} \mathbf{P}\left[F_<(x) = 2^i\epsilon(x)\right] \\
&= \sum_{i=2}^{i_{\max}} \mathbf{P}\left[B_{2^i\epsilon(x)}(x) < \left(1 - \frac{1}{e}\right) k \bigcap_{l=i+1}^{i_{\max}} B_{2^l\epsilon(x)}(x) \geq \left(1 - \frac{1}{e}\right) k\right] \\
&\leq \sum_{i=2}^{i_{\max}} \mathbf{P}\left[B_{2^i\epsilon(x)}(x) < \left(1 - \frac{1}{e}\right) k\right].
\end{aligned}$$

Furthermore we have:

$$\mathbf{E}\left[B_{\epsilon(x)2^i}(x)\right] = k - k(1 - \epsilon(x)2^i)^{N(x)} > \left(1 - \left(\frac{1}{e}\right)^{2^{i-1}}\right) k > \left(1 - \frac{1}{e}\right) k,$$

where the first equality follows from Lemma 1, while the second inequality follows recalling that $\epsilon(x) \geq 1/2N(x)$ and the applying Fact 2 with some straightforward manipulations. As a consequence, if we set

$$\delta_i = \frac{\mathbf{E}\left[B_{\epsilon(x)2^i}(x)\right] - \left(1 - \frac{1}{e}\right) k}{\mathbf{E}\left[B_{\epsilon(x)2^i}(x)\right]} = \frac{\frac{1}{e} - (1 - \epsilon(x)2^i)^{N(x)}}{1 - (1 - \epsilon(x)2^i)^{N(x)}},$$

we have $0 < \delta_i < 1$ and the event $(B_{\epsilon(x)2^i}(x) < (1 - 1/e)k)$ implies $(B_{\epsilon(x)2^i}(x) < (1 - \delta_i)\mathbf{E}\left[B_{\epsilon(x)2^i}(x)\right])$, where $B_{\epsilon(x)2^i}(x)$ is the sum of independent, binary random variables. Hence, we can apply Chernoff's bound [Mitzenmacher and Upfal 2005] to obtain:

$$\begin{aligned}
&\mathbf{P}\left[B_{\epsilon(x)2^i}(x) < (1 - 1/e)k\right] \leq e^{-\frac{\delta_i^2 \mathbf{E}\left[B_{\epsilon(x)2^i(x)}\right]}{2}} = e^{-\frac{\left(\frac{1}{e} - (1 - \epsilon(x)2^i)^{N(x)}\right)^2}{1 - (1 - \epsilon(x)2^i)^{N(x)}} k} \\
&\leq e^{-\frac{\left(\frac{1}{e} - \left(\frac{1}{e}\right)^{2^{i-1}}\right)^2}{2} k} \leq e^{-\frac{\left(\frac{1}{e} - \frac{1}{e^2}\right)^2}{2} k}.
\end{aligned}$$

The third inequality follows recalling that $\epsilon(x) \leq 1/2N(x)$ and applying Fact 2, while the fourth follows since $i \geq 2$. As a consequence:

$$\begin{aligned}
&\mathbf{P}\left[\overline{N}(x) < (1/3)N(x)\right] \leq \sum_{i=2}^{i_{\max}} \mathbf{P}\left[F_<(x) = 2^i\epsilon(x)\right] \leq \sum_{i=2}^{i_{\max}} e^{-\frac{\left(\frac{1}{e} - \frac{1}{e^2}\right)}{2} k} \\
&\simeq \sum_{i=2}^{i_{\max}} e^{-0.027k} = (\log_2 N)e^{-0.027k}.
\end{aligned}$$

We now turn to $\mathbf{P}\left[\overline{N}(x) > 3N(x)\right]$. First note that $(\overline{N}(x) > 3N(x))$ is equivalent to $(F_<(x) < 1/3N(x))$, by the way $\overline{N}(x)$ is chosen and by the definition of $F_<(x)$.

In the analysis, we have to distinguish the cases $\epsilon(x) < 2/3N(x)$ and $\epsilon(x) \geq 2/3N(x)$. In the former case we write:

$$\begin{aligned}
&\mathbf{P}\big[\overline{N}(x) > 3N(x)\big] = \mathbf{P}\left[F_<(x) < \frac{1}{3N(x)}\right] \leq \mathbf{P}\left[F_<(x) < \frac{2\epsilon(x)}{3}\right] \\
&= \mathbf{P}[F_<(x) \leq \epsilon(x)/2] = \mathbf{P}\left[\bigcap_{i=0}^{i_{\max}} B_{\epsilon(x)2^i}(x) > \left(1 - \frac{1}{e}\right)k\right] \\
&\leq \mathbf{P}\left[B_{\epsilon(x)}(x) > \left(1 - \frac{1}{e}\right)k\right],
\end{aligned}$$

where the first equality follows from the definitions of $\overline{N}(x)$ and $F_<(x)$, the second inequality follows since $1/N(x) \leq 2\epsilon(x)$ by definition of $\epsilon(x)$, while the third equality is a consequence of the fact that, by the algorithm, the largest possible value for $F_<(x)$ that is smaller than $2\epsilon(x)/3$ is $\epsilon(x)/2$. Now, we have

$$\begin{aligned}
&\mathbf{E}\big[B_{\epsilon(x)}(x)\big] = k - k(1-\epsilon(x))^{N(x)} \leq k - k\left(1 - \frac{2}{3N(x)}\right)^{N(x)} \\
&\leq k - k\left(1 - \frac{1}{N(x)+1}\right)^{N(x)} < \left(1 - \frac{1}{e}\right)k,
\end{aligned}$$

where the first equality follows from Lemma 1, the second inequality follows since $\epsilon(x) < 2/3N(x)$, while the fourth follows from Fact 2. Now set

$$\delta = \frac{\left(1 - \frac{1}{e}\right)k - \mathbf{E}\big[B_{\epsilon(x)}(x)\big]}{\mathbf{E}\big[B_{\epsilon(x)}(x)\big]} = \frac{(1-\epsilon(x))^{N(x)} - \frac{1}{e}}{1 - (1-\epsilon(x))^{N(x)}},$$

where obviously $\delta < 1$. We can write:

$$\begin{aligned}
\mathbf{P}\big[\overline{N}(x) > 3N(x)\big] \;&\leq\; \mathbf{P}\left[B_{\epsilon(x)}(x) > \left(1 - \frac{1}{e}\right)k\right] \\
&\leq \mathbf{P}\big[B_{\epsilon(x)}(x) > (1+\delta)\mathbf{E}\big[B_{\epsilon(x)}(x)\big]\big] \leq e^{-\frac{\delta^2}{3}B_{\epsilon(x)}(x)} \\
&= e^{-\frac{\left((1-\epsilon(x))^{N(x)} - \frac{1}{e}\right)^2}{3(1-(1-\epsilon(x))^{N(x)})}k},
\end{aligned}$$

where the third inequality follows from the application of Chernoff bound. On the other hand, recalling that $\epsilon(x) < 2/3N(x)$ we get:

$$\frac{\left((1-\epsilon(x))^{N(x)} - \frac{1}{e}\right)^2}{3(1-(1-\epsilon(x))^{N(x)})}k \geq \frac{\left((1 - \frac{2}{3N(x)})^{N(x)} - \frac{1}{e}\right)^2}{3(1-(1-\frac{2}{3N(x)})^{N(x)})}k \geq 0.012\,k,$$

whenever $N(x) \geq 10$. In deriving the second inequality, we use use Fact 1 with $\beta = 3/2$ to conclude that $(1-2/3N(x))^{N(x)}$ achieves its minimum when $N(x) = 10$.

We now consider the case $\epsilon(x) \geq 2/3N(x)$. Proceeding the same way as before we get:

$$\mathbf{P}\big[\overline{N}(x) > 3N(x)\big] = \mathbf{P}[F_<(x) \leq \epsilon(x)/4] = \mathbf{P}\left[\bigcap_{i=-1}^{i_{\max}} B_{\epsilon(x)2^i}(x) > \left(1 - \frac{1}{e}\right) k\right]$$
$$\leq \mathbf{P}\left[B_{\epsilon(x)/2}(x) > \left(1 - \frac{1}{e}\right) k\right].$$

where $i_{\max}$ has been defined previously. Here, the first equality follows since $(\overline{N}(x) > 3N(x))$ is equivalent to $(F_<(x) < 1/3N(x))$ and the latter implies $(F_<(x) < \epsilon(x)/2)$ since we are assuming $\epsilon(x) \geq 2/3N(x)$. Proceeding as in the previous case, it is easy to prove that $\mathbf{E}\big[B_{\epsilon(x)/2}(x)\big] < (1 - 1/e)k$. We can then define:

$$\delta = \frac{\left(1 - \frac{1}{e}\right) k - \mathbf{E}\big[B_{\epsilon(x)/2}(x)\big]}{\mathbf{E}\big[B_{\epsilon(x)/2}(x)\big]} = \frac{(1 - \epsilon(x)/2)^{N(x)} - \frac{1}{e}}{1 - (1 - \epsilon(x)/2)^{N(x)}},$$

where obviously $\delta < 1$. Finally,

$$\mathbf{P}\left[B_{\epsilon(x)/2}(x) > \left(1 - \frac{1}{e}\right) k\right] \leq \mathbf{P}\big[B_{\epsilon(x)/2}(x) > (1 + \delta)\mathbf{E}\big[B_{\epsilon(x)/2}(x)\big]\big]$$
$$\leq e^{-\frac{\delta^2}{3} B_{\epsilon(x)/2}(x)} \leq e^{-0.043k},$$

where the third inequality follows by considering the expression of $\delta^2 \mathbf{E}\big[B_{\epsilon(x)/2}(x)\big]$, recalling that $\epsilon(x) \leq 1/N(x)$ by definition and applying Fact 1 to $(1 - 1/2N(x))^{N(x)}$ with the assumption that $N(x) \geq 10$. We therefore conclude:

$$\mathbf{P}\left[\overline{N}(x) > 3N(x) \bigcup \left(\overline{N}(x) > \frac{N(x)}{3}\right)\right] \leq \log_2 N(x) e^{-0.027k} + e^{-0.012k}.$$

□

Received MM 2007; Reviewed MM YYYY; Accepted MM YYYY.