Proceedings of the First International Workshop on

Adversarial Information Retrieval on the Web

AIRWeb 2005

Held in conjunction with the 14th International World Wide Web Conference, 10 May 2005, Chiba, Japan.

Edited by

Brian D. Davison

Department of Computer Science and Engineering,
Lehigh University, Bethlehem, PA USA

Sponsored by

Technical Report LU-CSE-05-030, Department of Computer Science and Engineering, Lehigh University, Bethlehem, PA, 18015 USA.

Contents

Welcome

Dear Participant:

Welcome to the First International Workshop on Adversarial Information Retrieval on the Web (AIRWeb). This workshop is intended to bring together researchers and practitioners that are concerned with the on-going efforts in adversarial information retrieval on the Web. We have a total of eight peer-reviewed papers to be presented -- five research presentations and three synopses of work in progress. All convey the latest results in adversarial web IR, and address topics such as web spam, blog spam, cloaking, redirection, link optimization for PageRank, automated link spam detection, link bombs, reverse engineering of ranking algorithms, and propaganda. In addition, we will have a panel session in which workshop participants may raise additional questions of interest to industry experts and researchers.

I extend my thanks to the authors and presenters, and to the members of the program committee for their work in contributing to the material that forms an outstanding first workshop. I also sincerely thank Ask Jeeves for their support of the workshop, enabling us to partially cover travel costs for many of the student presenters. It is my hope that you will find the work presented interesting enough that you will ask questions, contribute ideas, and perhaps get involved in future work in this area.

Brian D. Davison, Program Chair
Lehigh University, Bethlehem, PA
19 April 2005

Overview

Search is the single most common application used on the Web. The attraction of hundreds of millions of searches per day provides significant incentive to content providers to do whatever necessary to rank highly in search engine results. The use of techniques that push rankings higher than they belong is often called spamming a search engine (or spamdexing). Such methods typically include textual as well as link-based techniques. Like e-mail spam, search engine spam is a form of adversarial information retrieval; the conflicting goals of accurate results of search providers and high positioning by content providers provides an interesting and real-world environment to study techniques in optimization, obfuscation, and reverse engineering, in addition to the application of information retrieval and classification.

The AIRWeb'05 workshop solicited technical papers on any aspect of adversarial information retrieval on the Web. Particular areas of interest included, but were not limited to:

- search engine spam and optimization,
- crawling the web without detection,
- link-bombing,
- reverse engineering of ranking algorithms,
- advertisement blocking, and
- web content filtering.

Papers addressing higher-level concerns (e.g., whether 'open' algorithms can succeed in an adversarial environment, whether permanent solutions are possible, etc.) were also welcome.

Authors were invited to submit **papers** and **synopses** in PDF format. We encouraged submissions presenting novel ideas and work in progress, as well as more mature work. Submissions were judged by multiple experts on relevance, significance, originality, clarity, and technical merit.

Workshop Program

9:00am **Welcome**
9:15am **Session 1**

- Web Spam Taxonomy
  Zoltán Gyöngyi, Stanford University
  Hector Garcia-Molina, Stanford University
- Synopsis: An Analysis of Factors Used in Search Engine Ranking
  Albert Bifet, Technical University of Catalonia
  Carlos Castillo, University of Chile
  Paul-Alexandru Chirita, L3S Research Center
  Ingmar Weber, Max-Planck-Institute for Computer Science
- Synopsis: Web Spam, Propaganda and Trust
  Panagiotis T. Metaxas, Wellesley College
  Joseph DeStefano, College of the Holy Cross

10:30am **Morning Break**
11:00am **Session 2**

- Pagerank Increase under Different Collusion Topologies
  Ricardo Baeza-Yates, ICREA-Univ. Pompeu Fabra and University of Chile
  Carlos Castillo, Universitat Pompeu Fabra
  Vicente López, Universitat Pompeu Fabra
- Cloaking and Redirection: A Preliminary Study
  Baoning Wu, Lehigh University
  Brian D. Davison, Lehigh University
- Synopsis: Optimal Link Bombs are Uncoordinated
  Sibel Adali, Renssalaer Polytechnic Institute
  Tina Liu, Renssalaer Polytechnic Institute
  Malik Magdon-Ismail, Renssalaer Polytechnic Institute

12:30pm **Lunch**
2:00pm **Expert Panel Session**

- **Moderator:**
  - *Brian D. Davison, Lehigh University*
- **Panelists:**
  - *Andrei Broder, IBM Research*
  - *Soumen Chakrabarti, IIT Bombay*
  - *David Cohn, Google*
  - *Tim Converse, Yahoo*
  - *Marc Najork, Microsoft Research*

3:30pm **Afternoon Break**

4:00pm **Session 3**

- Blocking Blog Spam with Language Model Disagreement
  Gilad Mishne, University of Amsterdam
  David Carmel, IBM Research
  Ronny Lempel, IBM Research
- SpamRank -- Fully Automatic Link Spam Detection
  András A. Benczúr, Hungarian Academy of Sciences (MTA SZTAKI) and Eötvös Univ.
  Károly Csalogány, Hungarian Academy of Sciences (MTA SZTAKI) and Eötvös Univ.
  Tamás Sarlós, Hungarian Academy of Sciences (MTA SZTAKI) and Eötvös University
  Máté Uher, Hungarian Academy of Sciences (MTA SZTAKI)

5:00pm **Discussion**

- Directions for future work
- Organization of AIRWeb '06

Workshop Organizers

Program Chair

Brian D. Davison, Lehigh University

Program Committee Members

Andrei Broder, IBM Research
David Carmel, IBM Research Haifa
Tim Converse, Yahoo
Nick Craswell, Microsoft Research Cambridge
Matt Cutts, Google
Dennis Fetterly, Microsoft Research
David Gibson, IBM Research
David Hawking, CSIRO
David D. Lewis, Independent Consultant
Mark Manasse, Microsoft Research
Kevin McCurley, IBM Research
Urban Mueller, Search.ch
Marc Najork, Microsoft Research
Jan Pedersen, Yahoo
Bernhard Seefeld, Search.ch
Baoning Wu, Lehigh University
Tao Yang, Ask Jeeves/Univ. of California, Santa Barbara

Contributing Authors

Sibel Adali, *Renssalaer Polytechnic Institute*
Ricardo Baeza-Yates, *ICREA-Univ. Pompeu Fabra and University of Chile*
András A. Benczúr, *Hungarian Academy of Sciences (MTA SZTAKI) and Eötvös University*
Albert Bifet, *Technical University of Catalonia*
David Carmel, *IBM Research*
Carlos Castillo, *University of Chile and Universitat Pompeu Fabra*
Paul-Alexandru Chirita, *L3S Research Center*
Károly Csalogány, *Hungarian Academy of Sciences (MTA SZTAKI) and Eötvös University*
Brian D. Davison, *Lehigh University*
Joseph DeStefano, *College of the Holy Cross*
Hector Garcia-Molina, *Stanford University*
Zoltán Gyöngyi, *Stanford University*
Ronny Lempel, *IBM Research*
Tina Liu, *Renssalaer Polytechnic Institute*
Vicente López, *Universitat Pompeu Fabra*
Malik Magdon-Ismail, *Renssalaer Polytechnic Institute*
Panagiotis T. Metaxas, *Wellesley College*
Gilad Mishne, *University of Amsterdam*
Tamás Sarlós, *Hungarian Academy of Sciences (MTA SZTAKI) and Eötvös University*
Máté Uher, *Hungarian Academy of Sciences (MTA SZTAKI)*
Ingmar Weber, *Max-Planck-Institute for Computer Science*
Baoning Wu, *Lehigh University*

Blocking Blog Spam with Language Model Disagreement

Gilad Mishne
Informatics Institute, University of Amsterdam
Kruislaan 403, 1098SJ Amsterdam
The Netherlands
gilad@science.uva.nl

David Carmel, Ronny Lempel
IBM Research Lab in Haifa
Haifa 31905, Israel
{carmel,rlempel}@il.ibm.com

ABSTRACT

We present an approach for detecting link spam common in blog comments by comparing the language models used in the blog post, the comment, and pages linked by the comments. In contrast to other link spam filtering approaches, our method requires no training, no hard-coded rule sets, and no knowledge of complete-web connectivity. Preliminary experiments with identification of typical blog spam show promising results.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval - search engine spam; I.7.5 [**Document Capture**]: Document analysis - document classification, spam filtering; K.4.1 [**Computers and Society**]: Public Policy Issues - abuse and crime involving computers, privacy

General Terms

Algorithms, Languages, Legal Aspects

Keywords

Comment spam, language models, blogs

1. INTRODUCTION

The growing popularity of internet search as a primary access point to the web has increased the benefits of achieving top rankings from popular search engines, especially for commercially-oriented web pages. Combined with the success of link analysis methods such as PageRank, this led to a rapid growth in link spamming – creating links which are "present for reasons other than merit" [6]. A well-known example of link spamming is link farms – link exchange programs existing for the sole purpose of boosting the link-based prestige of participating sites; these farms are fairly easily identified by topological analysis. However, in recent years search engines are facing a new link spam problem which is harder to track down: *comment spam*.

Comment spam is essentially link spam originating from comments and responses added to web pages which support dynamic user editing. With the massive increase in the number of blogs in recent years, such pages have proliferated; additional editable web pages which are often the target of comment spammers are wikis [1] and guestbooks. Blogs have made the life of comment spammers easy: instead of setting up complex webs of pages linking to the spam page, the spammer writes a simple agent that visits random blog and wiki pages, posting comments that link back to her page. Not only is spamming easier, but the spammer also benefits from the relatively high prestige that many blogs enjoy, stemming both from the rapid content change in them and the density of links between them. Comment spam, and link spam in general, poses a major challenge to search engines as it severely threatens the quality of their ranking. Commercial engines are seeking new solutions to this problem [9]; accordingly, the amount of research concerning link spam is increasing [8, 2, 6].

In this paper, we follow a language modeling approach for detecting link spam in blogs and similar pages. Our intuition is simple: we examine the use of language in the blog post, a related comment, and the page linked from the comment. In the case of comment spam, these language models are likely to be substantially different: the spammer is usually trying to create links between sites that have no semantic relation, e.g., a personal blog and an adult site. We exploit this divergence in the language models to effectively classify comments as spam or non-spam. Our method can be deployed in two modes: in retrospective manner, when inspecting a blog page which already has comments (this is particularly useful for crawlers); or in an online manner, to be used by the blog software to block spammers on the fly.

The rest of the paper is organized as follows. In Section 2 we survey existing work in the area of link spam. Our language modeling approach is presented and formalized in Section 3. Section 4 follows with a description of preliminary experiments conducted using this method; we conclude in Section 5. This paper discusses ongoing work and is meant to provide the conceptual framework and initial results, rather than a complete analysis of our proposed solution.

2. RELATED WORK

2.1 Comment Spam

Most approaches to preventing comment spam are technical in nature, and include:

- Requiring registration from comment posters;

[1]Collaborative websites whose content can be edited such as Wikipedia – http://wikipedia.org

AIRWeb'05, May 10, 2005, Chiba, Japan.

- Requiring commenters to solve a *captcha* – a simple Turing test mechanism [17];
- Preventing HTML in comments;
- Preventing comments on old blog posts;
- Using lists of forbidden or allowed IP addresses for commenters ("blacklists" and "whitelists");
- Using internal redirects instead of external links in comments;
- Limiting the number (or rate) of comments being added ("throttling").

While some of these mechanisms can be quite effective, they also have disadvantages. Methods which require user effort such as registration reduce the number of spontaneous responses which are important to many blog maintainers. Additionally, they do not affect the millions of commented web pages already "out there", and only address new comments. Preventing commenting altogether, or limiting it to plain text, or enforcing redirects on links in it, limits also legitimate comments and links contained in them, reducing the effectiveness of link analysis methods. Blacklists and whitelists require constant maintenance, and are bypassed by spammers using proxies and spoofed legitimate IP addresses. Throttling can reduce the amount of spam from a single page, but not the phenomenon altogether; spammers will simply post to more blogs.

Recently, a number of major search engines such as Yahoo, MSN Search and Google announced that they are collaborating with blogging software vendors and hosts to fight comment spam using a special attribute added to hypertext links [13]. This tag, `rel="nofollow"`, tells search engines that the links are untrusted, and will effectively prevent application of link-analysis scores to links associated with it – maybe even prevent crawling them altogether. However, the usage of this attribute is problematic for a number of reasons, including harming the inter-linked nature of blogs and possible abuse by webmasters; indeed, it is disputed within the blogging community [14, 4], and many do not intend to use it (e.g., at the time of writing, Yahoo's own search blog – which announced the tag – does not implement it).

2.2 Content Filtering and Spam

A different set of approaches for fighting comment spam works by analyzing the content of the spam comment, and possibly also the contents of pages linked by the comment (e.g., [12]). All these techniques are currently based on detecting a set of keywords or regular expressions within the comments. This approach suffers from the usual drawbacks associated with a manual set of rules, i.e. high maintenance load as spammers are getting more sophisticated. Typically, content-based methods require training with a large amount of spam and non-spam text, and correcting mistakes that are made; failure to continuously maintain the learner will decrease its accuracy, as it will create an inaccurate conception of what's spam and what's not. Having said that, regular expression based methods are fairly successful currently, partly due to the relatively young history of link spamming. It is expected that, similarly to the email spam world, as comment spammers enhance their methods, rule-based approaches will become less effective.

Published work on spam refers mostly to email spam, which was popular long before comment spam was, and was therefore targeted from many industrial and academic angles. In the email domain, machine learning and language modeling approaches have been very effective in classifying spam [10, 3]. An important difference between email spam and comment spam stems from the fact that comment spam is *not intended for humans*. No comment spammer actually expects anyone to click on the link that was added: this link is meant solely for the purpose of being followed by web crawlers. Thus, the spammer can (and does) use any type of words/features in his comment: the main goal is to have the link taken into account by search engine ranking schemes, and strings which have been reported as good discriminators of email spam such as over-emphasized punctuation [16] are not necessarily typical of comment spam.

2.3 Identifying Spam Sites

An altogether different approach to spam filtering is not to classify individual links as spam links or legitimate links, but to classify pages or sites as spam; recent work in this area includes usage of various non-content features [8] and link analysis methods [2]. The drawback of these approaches is that spam is essentially not a feature of pages, but of links between pages; sites can have both legitimate and spam incoming links (this is true for many online shops). Additionally, usage of some of these methods requires full connectivity knowledge of the domain, which is beyond the abilities of most bloggers.

In comparison to the existing methods presented, our approach requires no training, no maintenance, and no knowledge of additional information except that present on the commented web page.

3. COMMENT SPAM AND LANGUAGE MODELS

In this section we outline our language model based approach to identifying comment spam.

In the previous section, we noted that email spam is easier to classify than comment spam since it tends to have characterizing features – features which are supposed to convince a human to respond to the spam mail. On the other hand, comment spam has an advantage (from the filtering perspective) that email spam does not have. While every email needs to be classified as spam in an isolated manner, blog comments are presented within a *context*: a concrete semantic model in which the comment was posted. Our main assumption is that spam comments are more likely to violate this context by presenting completely different issues and topics. We instantiate the semantic models of the context, the comment and the page linked by the comment using language models.

3.1 Language Models for Text Comparison

A language model is a statistical model for text generation: a probability distribution over strings, indicating the likelihood of observing these strings in a language. Usually, the real model of a language is unknown, and is estimated using a sample of text representative of that language. Different texts can then be compared by estimating models for each of them, and comparing the models using well-known methods for comparing probability distributions. Indeed, the use of language models to compare texts in the Information Retrieval setting is empirically successful and becoming

increasingly popular [15, 11].

As noted earlier, we identify three types of languages, or language models, involved in the comment spam problem (see Figure 1). First, there is the model of the original blog post. Then, every comment added to the post adds two more models: the language used in the comment, and the language used in the page linked by the comment.

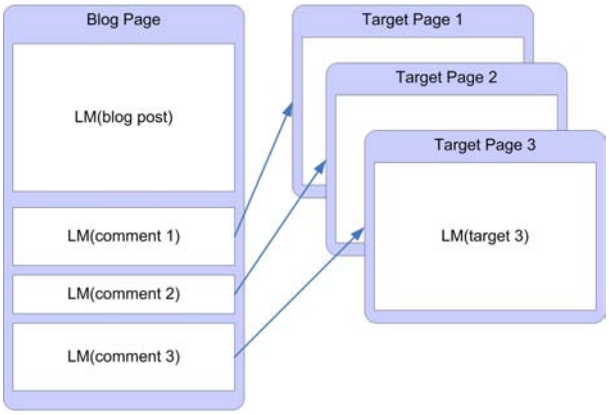


Figure 1: Three types of language models: the model of the blog post, the models of the comments, and the models of the pages linked by the comments.

To compare the models, we follow a variation on the Interpolated Aggregate Smoothing used by Allan *et. al.* in [1]. In practice, this measure calculates the smoothed Kullback-Leibler divergence between the language model of a short fragment of text and a combined language model of knowledge preceding this text. Formally, the KL-divergence between two probability distributions Θ_1, Θ_2 is

$$KL(\Theta_1 \| \Theta_2) = \sum_w p(w|\Theta_1) \log \frac{p(w|\Theta_1)}{p(w|\Theta_2)}$$

where $p(w|\Theta_i)$ is the probability of observing the word w according to the model Θ_i. In Interpolated Aggregate Smoothing, probabilities are estimated using maximum likelihood models and smoothed with Jelinek-Mercer smoothing. The two language models we are comparing are any pair of the triplet (blog post, comment, page linked by commet); let's examine the comparison between the model of the blog post (Θ_P), and the model of a comment to this post (Θ_C). We estimate the probabilities using maximum likelihood and smooth using a general probability model of words on the internet obtained from [7]. This gives us:

$$\begin{aligned} p(w|\Theta_P) &= \lambda_1 p(w|\Theta_{ML(post)}) \\ &+ (1-\lambda_1) p(w|\Theta_{ML(internet)}) \end{aligned}$$

$$\begin{aligned} p(w|\Theta_C) &= \lambda_2 p(w|\Theta_{ML(comment)}) \\ &+ (1-\lambda_2) p(w|\Theta_{ML(internet)}) \end{aligned}$$

Where $ML(\langle source \rangle)$ are maximum likelihood estimates. The language models used here are unigram models, but it is possible to use n-grams of higher orders in the same manner.

3.2 Spam Classification

Once we have a language model for each comment and a score based on its similarity to the language model of the blog post, we use these scores to classify the comment as spam or non-spam. Although this can be done using a simple threshold, we follow a different approach. We assume that the spammed blog page is parsed by a crawler, and the crawler is trying to assess which links (from the comments) should be used for link analysis methods and which not. In this case, the crawler views a range of comments, and must distinguish the good ones from the bad. As a first stage, the crawler calculates the KL-divergence for all comments as indicated above. The values obtained can then be seen as drawn from a probability distribution which is a mixture of Gaussians: each Gaussian represents a different language model. The Gaussian with the lowest mean – the least distance from the language model of the original blog post – represents the language model which is closest to the original post. Subsequent Gaussians represent language models which have a larger deviation from the original one, and are therefore more likely to constitute spam comments.

For our model, we assume the KL-divergence scores to be drawn from a 2-Gaussian distribution: the "legitimate" language model, and all other (spam) models (see example of the distribution in one of the blog pages in Figure 2). To estimate the parameters of the Gaussians, we use the EM algorithm.

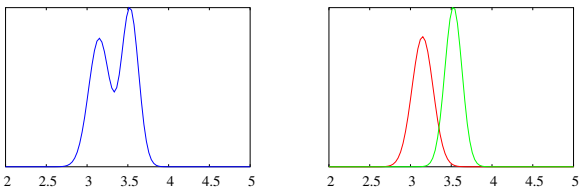


Figure 2: Gaussian mixture model estimated from the KL-divergence values of 10 comments on a blog post, and its underlying Gaussians

Finally, a comment is classified as spam if its KL-divergence from the blog post is more likely to be drawn from the spam Gaussian than from the legitimate one. For this purpose, we calculate a discriminating value between the two Gaussians – a number for which lower values are more likely to be drawn from the Gaussian with lower mean, and higher values are more likely to be drawn from the other Gaussian. Visually, this threshold can be viewed as the best vertical separator between the two distributions. Note that this threshold value provides us with an easy mechanism for changing the likelihood of identifying false positives ("legitimate" comments classified as spam) and false negatives (unidentified spam). Decreasing the threshold ("moving" the separator left) will result in a more strict requirement from the language model divergence between the comment and the post, effectively increasing the number of false positives and reducing false negatives; increasing the threshold value ("moving" the line to the right) will cause our method to be more tolerant to higher KL-divergence values, reducing false positives at the cost of increased false negatives. Usually, the cost of false positives is considered higher than that of false negatives; in general, we can use a *threshold multiplier* value to adjust the original threshold (where a multiplier of 1.0 will leave the threshold unchanged).

3.3 Model Expansion

Blog comments can be very short, and this is true also for some blog posts. This results in sparse language models, containing a relatively low number of words. We therefore propose to enrich the models of both the post and the comment, to achieve a more accurate estimation of the language model. An intuitive way to do so is to follow links present in the post and the comment, and add their content to the post and the comment, respectively; in the case of the post, it is also possible to follow incoming links to the blog and add their content. Taking this a step further, it is also possible to continue following links up to depth N, although this potentially causes topic (and language model) drift.

3.4 Limitations and Solutions

An easy way for spammers to "cheat" our model (or any other model which compares the contents of the post and the comment) is to generate comments with a similar language model to the original blog post. This makes the link-spam bots slightly more complicated since they must identify the post contents and use its model for generating a close one (e.g., by copying phrases) – but spammers have shown to overcome much higher obstacles.

However, in this case of language model faking, a new opportunity arises: assuming the spammer posts multiple comments in many different blogs (as a means of increasing the PageRank), there are now many comments with completely different language models to the same spam site. This is easily detectable at the search engine level which has a connectivity server at its hand; it can also be detected by an iterative HITS-like method by the blogger, following the link to the spam site and then its incoming links.

As any other spam filtering method, ours is not foolproof and can make mistakes; comments which are formulated with a sufficiently different vocabulary than the blog post might be mistaken for spam. However, this is precisely the reason for the robustness of our approach: it is very hard for the spammer to create comments that will be both similar to the blog language and to the spam site language. To account for these mistakes, an alternative to using our method as a binary spam/non-spam classifier is to use it for assigning a weight to links found in comments according to their language model divergence; the weight can be used to decrease PageRank of malicious sites, using methods such as the one reported in [5].

4. EXPERIMENTAL SETTING

We now present some preliminary experiments in identifying comment spam in blogs using our approach.

We collected 50 random blog posts, along with the 1024 comments posted to them; all pages contain a mix of spam and non-spam comments. The number of comments per post ranged between 3 and 96, with the median being 13.5 (duplicate and near-duplicate comments were removed). We manually classified the comments: 332 (32%) were found to be "legitimate" comments, some of them containing links to related pages and some containing no links; the other 692 comments (68%) were link-spam comments [2].

The link-spam comments we encountered in our corpus are of diverse types; while some of them are simple keyword lists, accompanied by links to the spam sites, others employ more sophisticated language (see Figure 3 for some sample comments). A typical blog page from our corpus contains a mix of different comment spam types.

```
6708 sports bettingonline sports bettingmarch
madnessbasketball bettingncaa bettingsports ...
```
Link: gambling site

```
%syn(Cool|Nice|Rulezz)% %syn(blog,|portal|site)%
hope to make %syn(my own|own weblog|my diary)%,
not worse than yours ;)
```
Link: adult site

```
A common mistake that people make when trying
to design something completely foolproof was to
underestimate the ingenuity of complete fools.
```
Link: pharmacy site

```
i was looking for plus size lingerie but google
sent me here
```
Link: fashion shop

Figure 3: Samples of comment spam in our collection (top to bottom): [1] keyword based, with a random number to prevent duplicate detection; [2] revealing internal implementation of the spamming agent; [3] using quotes – in this case, from Douglas Adams – as "trusted" language; [4] disguising as random surfer

In this set of experiments, we compared only the language models of the post and the comments, and did not take into account the model of the page linked by the comments. This was done due to time constraints. The values of λ_i (see previous section) were both set to 0.9.

4.1 Results

The results of our experiments are shown in Table 1. False negatives are spam comments which were not identified as spam by our method, while false positives are non-spam comments which were classified as spam. The threshold multiplier is the value used to modify the separator between the two language models as described in section 3.2.

As a naive baseline, we use the maximum likelihood probabilities for the comment type in our model; as noted earlier, 68% of the comments were spam, so we assume an ad-hoc fixed probability of 0.68 for a comment to contain link spam. We achieve reasonable performance with our model, and can clearly see the trade-off between misclassifying spam and misclassifying non-spam, resulting from different modifications to the language model threshold.

Discussion. The size of our corpus is far from being satisfactory: we therefore label our results as a proof-of-concept and basis for continued experimentation, rather than full-fledged evidence of the method's capabilities. Nevertheless, the results are encouraging and clearly show that our in-

[2]The collection can be obtained from http://ilps.science.uva.nl/Resources/blogspam/

Table 1: Blog link-spam classification results

Method	Threshold Multiplier	Correct	False Negatives	False Positives
`Baseline (avg. 100 runs)`	N/A	581 (57%)	223 (21.5%)	220 (21.5%)
`KL-divergence`	0.75	840 (82%)	65 (6.5%)	119 (11.5%)
`KL-divergence`	0.90	834 (81.5%)	69 (6.5%)	121 (12%)
`KL-divergence`	1.00	823 (80.5%)	88 (8.5%)	113 (11%)
`KL-divergence`	1.10	850 (83%)	87 (8.5%)	87 (8.5%)
`KL-divergence`	1.25	835 (81.5%)	111 (11%)	78 (7.5%)

tuition is correct: the language used in spam comments does diverge from the language of the blog post substantially more than the language used in legitimate comments.

An analysis of the misclassified comments reveals that many of them are very short – containing 3-4 words, usually a non-content response to the post (e.g., "That sounds cool"). However, the vast majority of these comments contain no external links, or an email link only – so their misclassification will not result in actual search engine spam (in the case of false negatives) and not change the "true" link-analysis prestige of pages (in the case of false positives). While it is possible to integrate language divergence with comment length and other features into a hybrid comment spam classification system, we focused on the language aspect only and did not explore usage of additional knowledge.

Model Expansions. As mentioned earlier, a possible solution to the sparseness of some of the blog posts is to expand the language model in various ways. We performed a limited amount of experiments involving such expansions, by following all links present in the blog post and adding the content present in the target pages to the content of the blog post, before estimating the language model. Of the 50 blog posts in our corpus, 31 posts had valid links to other pages (some posts did not contain links at all, and some contained expired and broken links). The average number of links followed (for the 31 pages with expansions) was 3.4. Unfortunately, using the expanded models did not improve the overall classification accuracy. In fact, while for some blog posts – most notably shorter ones – the expansion helped substantially, we experienced a degradation of 2%-5% in the average performance over the entire corpus. However, both the fairly small number of pages which were expanded and the limited experiments performed prevent us from formulating a definite statement regarding model expansion at this stage.

5. CONCLUSIONS

We presented an approach for classifying blog comment spam by exploiting the difference between the language used in a blog post and the language used in the comments to that post (and pages linked from those comments). Our method works by estimating language models for each of these components, and comparing these models using well-known methods. Preliminary experiments using our method to classify typical comment spam show promising results; while in this paper we discuss blogs, the problem and the solution are relevant to other types of comment spam, such as wiki spam.

6. REFERENCES

[1] J. Allan, C. Wade, and A. Bolivar. Retrieval and novelty detection at the sentence level. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 314–321. ACM Press, 2003.

[2] E. Amitay, D. Carmel, A. Darlow, R. Lempel, and A. Soffer. The connectivity sonar: detecting site functionality by structural patterns. In *HYPERTEXT '03: Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, pages 38–47. ACM Press, 2003.

[3] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, and C. D. Spyropoulos. An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 160–167. ACM Press, 2000.

[4] Nofollow tag cheers bloggers, but fails blogs, URL: `http://www.platinax.co.uk/news/archives/2005/01/new_nofollow_ta.html`.

[5] R. Baeza-Yates and E. Davis. Web page ranking using link attributes. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 328–329. ACM Press, 2004.

[6] B. Davison. Recognizing nepotistic links on the web. In *AAAI-2000 workshop on Artificial Intelligence for Web Search*, pages 23–28. AAAI Press, 2000.

[7] The Berkeley/Stanford Web Term Document Frequency and Rank project, URL: `http://elib.cs.berkeley.edu/docfreq/`.

[8] D. Fetterly, M. Manasse, and M. Najork. Spam, damn spam, and statistics: using statistical analysis to locate spam web pages. In *WebDB '04: Proceedings of the 7th International Workshop on the Web and Databases*, pages 1–6. ACM Press, 2004.

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

[10] J. M. G. Hidalgo. Evaluating cost-sensitive unsolicited bulk email categorization. In *SAC '02: Proceedings of the 2002 ACM symposium on Applied computing*, pages 615–620. ACM Press, 2002.

[11] V. Lavrenko and W. B. Croft. Relevance based language models. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*,

pages 120–127. ACM Press, 2001.

[12] Movable Type Blacklist Filter, with content filtering, URL: http://www.jayallen.org/projects/mt-blacklist/.

[13] Joint statement from Yahoo, Google, and others regarding the "nofollow" tag, URLs: http://www.google.com/googleblog/2005/01/preventing-comment-spam.html, http://www.ysearchblog.com/archives/000069.html.

[14] No nofollow: fight spam, not blogs, URL: http://www.nonofollow.net.

[15] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281. ACM Press, 1998.

[16] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A bayesian approach to filtering junk E-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.

[17] L. von Ahn, M. Blum, and J. Langford. Telling humans and computers apart automatically. *Commun. ACM*, 47(2):56–60, 2004.

Cloaking and Redirection: A Preliminary Study

Baoning Wu and **Brian D. Davison**
Computer Science & Engineering
Lehigh University
{baw4,davison}@cse.lehigh.edu

Abstract

Cloaking and redirection are two possible search engine spamming techniques. In order to understand cloaking and redirection on the Web, we downloaded two sets of Web pages while mimicking a popular Web crawler and as a common Web browser. We estimate that 3% of the first data set and 9% of the second data set utilize cloaking of some kind. By checking manually a sample of the cloaking pages from the second data set, nearly one third of them appear to aim to manipulate search engine ranking.

We also examined redirection methods present in the first data set. We propose a method of detecting cloaking pages by calculating the difference of three copies of the same page. We examine the different types of cloaking that are found and the distribution of different types of redirection.

1 Introduction

Cloaking is the practice of sending different content to a search engine than to regular visitors of a web site. Redirection is used to send users automatically to another URL after loading the current URL. Both of these techniques can be used in search engine spamming [13, 7]. Henzinger et al. [8] has pointed out that search engine spam is one of the major challenges of web search engines and cloaking is among the spamming techniques used today. Since search engine results can be severely affected by spam, search engines typically have policies against cloaking and some kinds of dedicated redirection [5, 16, 1].

Google [5] describes cloaking as the situation in which "the webserver is programmed to return different content to Google than it returns to regular users, usually in an attempt to distort search engine rankings." An obvious solution to detect cloaking is that for each page, calculate whether there is a difference between a copy from a search engine's perspective and a copy from a web browser's perspective. But in reality, this is non-trivial. Unfortunately, it is not enough to know that corresponding copies of a page differ; we still cannot tell whether the page is a cloaking page. The reason is that web pages may be updated frequently, such as in a news website or a blog website, or simply that the web site puts a time stamp on every page it serves. Even if two crawlers were synchronized to visit the same web page at nearly the same moment, some dynamically generated pages may still have different content, such as a banner advertisement that is rotated on each access.

Besides the difficulty of identifying cloaking, it is also hard to tell whether a particular instance of cloaking is considered acceptable or not. We define the cloaking behavior that has the effect of manipulating search engine ranking results as semantic cloaking. Unfortunately, the various search engines may have different criteria for defining unacceptable cloaking. As a result, we have focused on the simpler, more basic task — when we mention cloaking in this paper, we usually refer to the simpler case of whether different content is served to automated crawlers versus web browsers, but not different content to every visitor. We name this cloaking as syntactic cloaking. So, for example, we will not consider dynamic advertisements to be cloaking.

In order to investigate this issue, we collected two data sets: one is a large data set containing 250,000 pages and the other is a smaller data set containing 47,170 pages. The detail of these two data set will be given in Section 3. We manually examined a number of samples of those pages and found several different kinds of cloaking techniques. From this study we make an initial proposition toward building an auto-

AIRWeb'05, May 10, 2005, Chiba, Japan.

mated cloaking detection system. Our hope is that these results may be of use to researchers to design better and more thorough solutions to the cloaking problem.

Since redirection can also be used as a spamming technique, we also calculated some statistics based on our crawled data for cloaking. Four types of redirection are studied.

Few publications address the issue of cloaking on the Web. As a result, the main contribution of this paper is to begin a discussion of the problem of cloaking and its prevalence in the web today. We provide a view of actual cloaking and redirection techniques. We additionally propose a method for detecting cloaking by using three copies of the same page.

We next review those few papers that mention cloaking. The data sets we use for this study are introduced in Section 3. The results of cloaking and redirection are shown in Section 4 and 5 respectively. We conclude this paper with a summary and discussion in Section 6.

2 Related Work

Henzinger et al. [8] mentioned that search engine spam is quite prevalent and search engine results would suffer greatly without taking measures. They also mentioned that cloaking is one of the major search engine spam techniques.

Gyöngyi and Garcia-Molina [7] describe cloaking and redirection as spam hiding techniques. They showed that web sites can identify search engine crawlers by their network IP address or user-agent names. They also described the use of refresh meta tags and JavaScript to perform redirection. They additionally mention that some cloaking (such as sending search engine a version free of navigational links, advertisements but no change to the content) are accepted by search engines.

Perkins [13] argues that agent-based cloaking is spam. No matter what kind of content is sent to search engine, the goal is to manipulate search engines rankings, which is an obvious characteristic of search engine spam.

Cafarella and Cutting [4] mention cloaking as one of the spamming techniques. They said that search engines will fight cloaking by penalizing sites that give substantially different content to different browsers.

None of the above papers discuss how to detect cloaking, which is one aspect of the present work. In one cloaking forum [14], many examples of cloaking and methods of detecting cloaking are proposed and discussed. Unfortunately, generally these discussions can be taken as speculation only, as they lack strong evidence or conclusive experiments.

Najork filed for patent [12] on a method for detecting cloaked pages. He proposed an idea of detecting cloaked pages from users' browsers by installing a toolbar and letting the toolbar send the signature of user perceived pages to search engines. His method does not distinguish rapidly changing or dynamically generated Web pages from real cloaking pages, which is a major concern for our algorithms.

3 Data set

Two data sets were examined for our cloaking and redirection testing. For convenience, we name the first data as HITSdata and the second as HOTdata.

3.1 First data set: HITSdata

In related work to recognize spam in the form of link farms [15], we collected Web pages in the neighborhood of the top 100 results for 412 queries by following the HITS data collection process [9]. That is, for each query presented to a popular search engine, we collected the top 200 result references, and for each URL we also retrieved the outgoing link set, and up to 100 incoming link pages. The resulting data set contains 2.1M unique Web pages. From these 2.1M URLs, we randomly selected 250,000 URLs. In order to test for cloaking, we crawled these pages simultaneously from a university IP address (Lehigh) and from a commercial IP address (Verizon DSL). We set the *user-agent* from the university address to be `Mozilla/4.0 (compatible; MSIE 5.5; Windows 98)` and the one from the commercial IP to be `Googlebot/2.1 (+http://www.googlebot.com/bot.html)`. From each location we crawled our dataset twice with a time interval of one day. So, for each page, we finally have four copies, two of which are from a web browser's perspective and two from a crawler's perspective. For convenience, we name these four copies as $B1$, $B2$, $C1$ and $C2$ respectively. For each page, the time order of retrieval of these four copies is always $C1$, $B1$, $C2$ and $B2$.

3.2 Second data set: HOTdata

We also want to know the cloaking ratio within the top response lists for hot queries.

The first step is to collect hot queries from popular search engines. To do this, we collected 10 popular queries of Jan 2005 from Google Zeigeist [6], top 100 search terms of 2004 from Lycos [10], top 10 searches for the week ending Mar 11, 2005 from Ask Jeeves [3], and 10 hot searches in each of 16 categories ending Mar 11, 2005 from AOL [2]. This resulted in 257 unique queries from these web sites.

The second step is to collect top response list for these hot queries. For each of these 257 queries, we retrieved the top 200 responses from the Google search engine. The number of unique URLs is 47,170. Like the first data set, we downloaded four copies for each of these 47,170 URLs, two from a browser's perspective and two from a crawler's perspective. But all these copies are downloaded from machines with a university IP address. For convenience, we name these four copies $HC1$, $HB1$, $HC2$ and $HB2$ respectively. This order also matches the time order of downloading them.

4 Results of Cloaking

In this section, we will show the results for the cloaking test.

4.1 Detecting Cloaking in HITSdata

Intuitively, the goal of cloaking is to give different content to a search engine than to normal web browsers. This can be different text or links. We use two techniques to compare versions retrieved by a crawler and a browser — we consider the number of differences in the terms and links used over time to detect cloaking.

As we mentioned earlier in Section 1, calculating the difference between pages from the browser's and crawler's viewpoints is not strong enough to tell whether the page does cloaking. Our proposed method is that we can use three copies of a page $C1$, $C2$ and $B1$ to decide if it is a cloaking page. The detail is that for each URL, we first calculate the difference between $C1$ and $C2$ (for convenience, we use NCC to represent this number). Then we calculated the difference between $B1$ and $C1$ (for convenience, we use NBC to represent this number). Finally if NBC is greater than NCC, then we mark it as a cloaking candidate. The intuition is that the page may change frequently, but if the difference between the browser's copy and the crawler's copy is bigger than the difference between two crawler copies, the evidence may be enough that the page is cloaking.

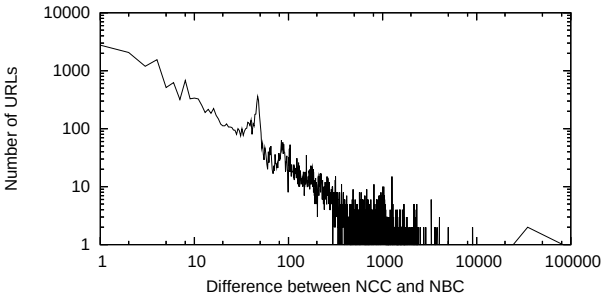


Figure 1: Distribution of the difference of NCC and NBC.

We used two methods to calculate the difference between pages — the difference in terms used, and the difference in links provided. We describe each below, along with the results obtained.

4.1.1 Term Difference

The first method for detecting cloaking is to use term difference among different copies. Instead of using all the terms in the HTML files, we used the "bag of words" method for analyzing the web pages, i.e., we parse the HTML file into terms and only count each unique term once no matter how many times this term appears. Thus, each page is marked by a set of words after parsing.

For each page, we first calculated the number of different terms between the copies $C1$ and $C2$ (designated NCC, as described above). We then calculated the number of different terms between the copies $C1$ and $B1$, (designated NBC). We then select pages that have a bigger NBC than NCC as candidates of cloaking. For this data set, we marked 23,475 candidates of the original 250K data set.

The distribution of the difference of these 23,475 pages forms a power-law-like distribution, shown in Figure 1.

To check what threshold for this difference between NCC and NBC is a good indication for real cloaking, first, we put the 23,475 URLs into ten different buckets based on the difference value. The range for each bucket and the number of pages within each bucket are shown in Table 1.

Then, from each bucket we randomly selected

Bucket ID	RANGE	No. of Pages
1	$x <= 5$	8084
2	$5 < x <= 10$	2287
3	$10 < x <= 20$	1938
4	$20 < x <= 40$	2065
5	$40 < x <= 80$	2908
6	$80 < x <= 160$	1731
7	$160 < x <= 320$	1496
8	$320 < x <= 640$	912
9	$640 < x <= 1280$	1297
10	$1280 < x$	757

Table 1: Buckets of term difference

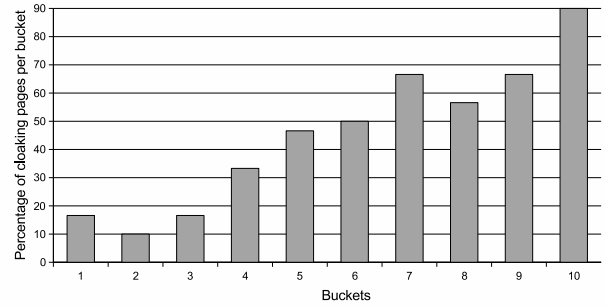


Figure 2: The ratio of syntactic cloaking in each bucket based on term difference.

thirty pages and checked them manually to see how many from these thirty pages are real syntactic cloaking pages within each bucket. The result is shown in Figure 2.

The trend is obvious in Figure 2. The greater the difference, the higher proportion of cloaking that is contained in the bucket. In order to know which is the optimal threshold to choose, we calculated the precision, recall and F-measure based on the range of these buckets. For these three measures, we follow the definitions in [11] and select α to be 0.5 in the F-measure formula to give equal weight to recall and precision. Precision is the proportion of selected items that the system got right; Recall is the proportion of the target items that the system selected; F-measure is the measure that combines precision and recall. The results of these three measures are shown in Table 2. If we choose F-measure as the criteria, buckets 4 and 5 have the highest value. Since the range of bucket 4 and 5 is around 40 in Table 1, we can set the threshold to be 40 and declare that all pages with the difference above 40 to be categorized as cloaking pages. In that case, the precision and

Threshold	PRECISION	RECALL	F value
1	0.355	1.000	0.502
5	0.423	0.828	0.560
10	0.480	0.799	0.560
20	0.534	0.758	0.627
40	0.580	0.671	0.622
80	0.633	0.498	0.588
160	0.685	0.388	0.496
320	0.695	0.262	0.380
640	0.752	0.196	0.311
1280	0.899	0.086	0.157

Table 2: F-measure for different thresholds based on term difference.

recall are 0.580 and 0.671 respectively.

From Figure 2, we can make an estimation of what percentage of our 250,000 page set are cloaking pages. Since we know the total number of pages within each bucket and the number of cloaking pages within the 30 manually checked pages from each bucket, the estimation of total number of cloaking pages is the product of the number of pages within each bucket and the ratio of cloaking pages within the 30 pages. The result is 7,780, so we expect that we can identify nearly 8,000 cloaking pages (about 3%) within the 250,000 pages.

4.1.2 Link Difference

Similar to term difference, we also analyzed this data sets on the basis of link differences. Here link difference means the number of different links between two corresponding pages.

First we calculated the link difference between the copy of $C1$ and $C2$ (termed LCC).We then calculated the link difference between the copy of $C1$ and $B1$ (termed LBC). Finally we marked the page that have a higher LBC than LCC as cloaking candidates. In this way, we marked 8,205 candidates. The frequency of these candidates also approximates a power-law distribution like term cloaking. It is shown in Figure 3.

As with term difference, we also put these 8,205 candidates into 10 buckets. The range and number of pages within each bucket is shown in Table 3.

From each bucket, we randomly selected 30 pages and checked manually to see how many of them are real cloaking pages. The result is shown in Figure 4.

It is obvious that the most of the pages from bucket 4 or above are cloaking pages. We also calculated the F values for these thresholds corresponding to the

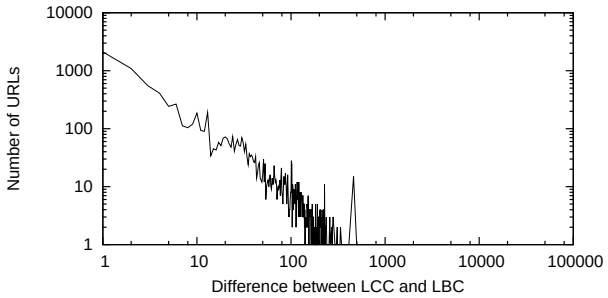


Figure 3: Distribution of the difference of LCC and LBC.

Bucket ID	RANGE	No. of Pages
1	$x <= 5$	4415
2	$5 < x <= 10$	787
3	$10 < x <= 20$	746
4	$20 < x <= 35$	783
5	$35 < x <= 55$	441
6	$55 < x <= 80$	299
7	$80 < x <= 110$	279
8	$110 < x <= 145$	182
9	$145 < x <= 185$	100
10	$185 < x$	173

Table 3: Buckets of link difference

range of each bucket. The result is shown in Table 4. We can tell that 5 is an optimal threshold with the best F value.

Since the number of pages having link difference is smaller than the ones having term difference in reality, fewer cloaking pages can be found by using link difference alone, but are more accurate.

Threshold	PRECISION	RECALL	F value
1	0.479	1.000	0.648
5	0.727	0.700	0.713
10	0.822	0.627	0.711
20	0.906	0.520	0.660
35	0.910	0.340	0.496
55	0.900	0.236	0.374
80	0.900	0.167	0.283
110	0.900	0.104	0.186
145	0.878	0.060	0.114
185	0.866	0.038	0.072

Table 4: F-measure for different thresholds based on link difference.

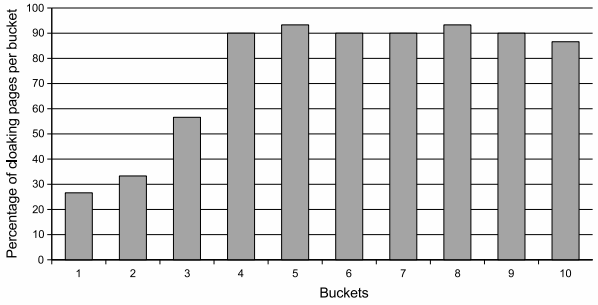


Figure 4: The ratio of syntactic cloaking in each bucket based on link difference.

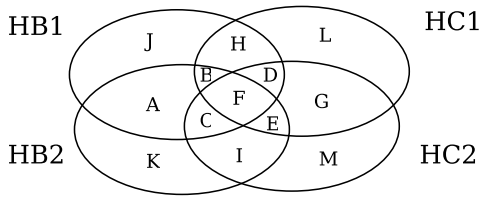


Figure 5: Intersection of the four copies for a Web page.

4.2 Detecting Cloaking in HOTdata

Based on the experience of manually checking for cloaking pages for the first data set, we attempted to detect syntactic cloaking automatically by using all four copies of each page.

4.2.1 Algorithm of detecting cloaking automatically

Our assumption about syntactic cloaking is that the web site will send something consistent to the crawler but send something different yet still consistent to the browser. So, if there exists such terms that only appear in both of the copies sent to the crawler but never appear in any of the copies send to the browser or vice versa, it is quite possible that the page is doing syntactic cloaking. Here when getting the terms out of each copy, we still use the "bag of words" approach, i.e., we replace all the non-word characters within an HTML file with blank and then get all the words out of it for the intersection operation.

To easily describe our algorithm, the intersection of four copies are shown as a Venn diagram in Fig-

Bucket	RANGE	No.	Accuracy
1	$x <= 1$	725	40%
2	$1 < x <= 2$	540	30%
3	$2 < x <= 4$	495	30%
4	$4 < x <= 8$	623	40%
5	$8 < x <= 16$	650	90%
6	$16 < x <= 32$	822	100%
7	$32 < x <= 64$	600	100%
8	$64 < x <= 128$	741	100%
9	$128 < x <= 256$	420	100%
10	$256 < x$	1120	100%

Table 5: Buckets of unique terms in area A and G

Thresholds	PRECISION	RECALL	F value
0	0.647	1.000	0.785
1	0.703	0.965	0.813
2	0.766	0.952	0.849
4	0.836	0.940	0.885
8	0.902	0.881	0.891
16	0.922	0.756	0.831
32	0.960	0.599	0.738
64	0.979	0.470	0.635
128	0.972	0.358	0.523
256	1.000	0.267	0.422

Table 6: F-measure of different threshold

ure 5. We use capital letters from A to M to represent each intersection component of four copies. For example, the area L contains content that only appears in $HC1$, but never appear in $HC2$, $HB1$ and $HB2$; area F is the intersection of four copies, i.e., the content that appears on all of the four copies. The most interesting components to us are areas A and G. Area A represents terms that appear on both browsers' copies but never appear on any of the crawlers' copies, while area G represents terms that appear on both crawlers' copies but never appear on any of the browsers' copies.

So our algorithm of detecting syntactic cloaking automatically is that for each web page, we calculate the number of terms in area A and the number of terms in area G. If the sum of these two numbers is nonzero, we may mark this page as a cloaking page.

There are false negative examples for this algorithm. A simple example is that suppose there is a dynamic picture on the page, every time the web server will randomly select one from 4 JPEG files (a1.jpg to a4.jpg) to serve the request. It happens that a1.jpg is sent every time when our crawler visits this page, but a2.jpg and a3.jpg are sent when our browser visit this page. By our algorithm, the page will be marked as cloaking, but it can be easily verified that this is not the case. So, again we need a threshold for the algorithm to work more accurately.

For the 47,170 URLs, we found 6466 pages that have the sum of number of terms in area A and G greater than 0. Again, we put them into 10 buckets, as shown in Table 5. The third column is the number of pages within this bucket.

From each bucket, we randomly selected 10 pages and manually checked to see whether this page is real syntactic cloaking. The accuracy is shown in the fourth column in Table 5. We also calculated the F-measure, the results are shown in Table 6.

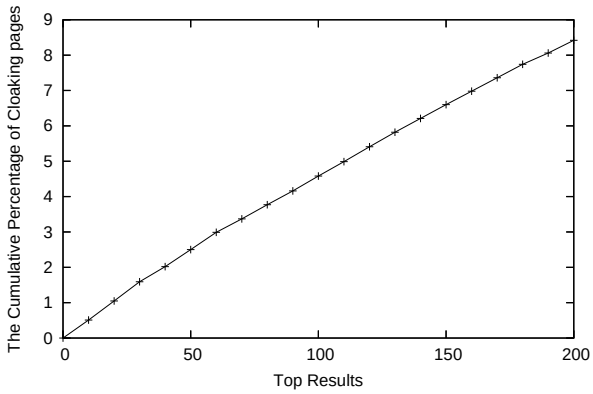
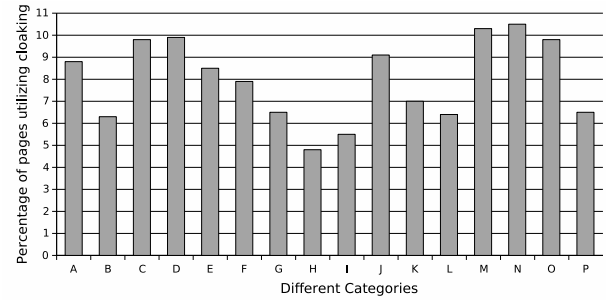


Figure 6: Percentage of syntactic cloaking pages within google's top responses.

Since the 4th and 5th bucket have highest F value in Table 6, we choose the threshold to be the range between bucket 4 and bucket 5, i.e., 8. So, our automated cloaking algorithm is revised to only mark pages with the sum of area A and G greater than 8 as cloaking pages. So, for our second data set, all pages in bucket 5 to bucket 10 are marked cloaking pages. Finally, we marked 4,083 pages out of the 47,170 pages, i.e., about 9% of pages from the hot query data set are syntactic cloaking pages.

4.2.2 Distribution of syntactic cloaking within top rankings

Since we have identified 4,083 pages that utilize cloaking, we can now draw the distribution of these cloaking pages within different top rankings. Figure 6 shows the cumulative percentage of cloaking pages within the Top 200 response lists returned by google. As we can see, about 2% of top 50, about 4% of top 100 URLs and more than 8% of top 200 URLs do utilize cloaking. The ratio is quite high and the cloaking



A. Autos	E. Games	I. Local	M. Shopping
B. Companies	F. Health	J. Movies	N. Sports
C. Computing	G. House	K. Music	O. TV
D. Entertainment	H. Holidays	L. Research	P. Travel

Figure 7: Category-specific Cloaking.

may be helpful for these pages to be ranked high.

Since we retrieved top 10 hot queries from each of 16 categories from AOL, we can consider the topic of the cloaking pages. Intuitively some popular categories, such as sports or computers, may contain more cloaking pages in the top ranking list. So we also calculated the fraction of cloaking pages within each category. The results are shown in Figure 7. Some categories, such as *Shopping* and *Sports*, are more likely to have cloaked results than other categories.

4.2.3 Syntactic vs Semantic cloaking

Not all syntactic cloaking is considered unacceptable to search engines. For example, a page sent to the crawler that doesn't contain advertising content or a PHP session identifier which is used to distinguish different real users is not a problem to search engines. In contrast to acceptable cloaking, we define semantic cloaking as cloaking behavior with the effect of manipulating search engine results.

To make one more step about our cloaking study, we randomly selected 100 pages from the 4,083 pages we have detected as syntactic cloaking pages and manually checked the percentage of semantic cloaking among them. In practice, it is difficult to judge whether some behavior is harmful to search engine rankings. For example, some web sites will send login page to browser, while send full page to crawler. So, we end up with three categories: acceptable cloaking, unknown and semantic cloaking.

From these 100 pages, we classified 33 pages as semantic cloaking, 32 as unknown and 35 as acceptable cloaking.

4.3 Different types of cloaking

In the process of manually checking 600 pages for the above sections, we found several different types of cloaking.

4.3.1 Types of term cloaking

We identified many different methods of sending different term content to crawlers and web browsers. They can be categorized by the magnitude of the difference.

We first consider the case in which the content of the pages sent to the crawler and web browser are quite different.

- The page provided to the crawler is full of detail, but the one to the web browser is empty, or only contains frames or JavaScript.
- The web site sends text page to the crawler, but sends non-text content (such as macromedia Flash content) to web browser.
- The page sent to the crawler incorporates content, but the one sent to the web browser contains only a redirect or 404 error response.

The second case is when content differs only partially between the pages sent to the crawler and the browser and the remaining content is identical, or one copy has slightly more content than the other.

- The pages sent to the crawler contain more text content than the ones to web browser. For example, only the page sent to the crawler contains keywords shown in Figure 8.
- Different redirection target URLs are contained in the pages sent to the crawler and to the web browser.
- The web site sends different titles, meta-description or keywords to the crawler than to web browser. For example, the header to browser uses "Shape of Things movie info at Video Universe" as the meta-description, while the one to the crawler uses "Great prices on Shape of Things VHS movies at Video Universe. Great service, secure ordering and fast shipping at everyday discount prices."
- The page sent to the crawler contains JavaScript, but no such JavaScript is sent to the browser, or the pages have different JavaScripts sent to the crawler than to web browser.

game computer games PC games console games video games computer action games adventure games role playing games simulation games sports games strategy games contest contests prize prizes game cheats hints strategy computer games PC games computer action games adventure games role playing games Nintendo Playstation simulation games sports games strategy games contest contests prize prizes game computer games PC games computer action games adventure games role playing games simulation games sports games strategy games contest contests prize prizes.

Figure 8: Sample of keywords content only sent to the crawler.

- Pages to the crawler do not contain some banner advertisements, while the pages to web browser do.
- The *NOSCRIPT* element is used to define an alternate content if a script is not executed. The page sent to web browser has the *NOSCRIPT* tag, while the page sent to the crawler does not.

4.3.2 Types of link cloaking

For link cloaking, we again group the situations by the magnitude of the differences between different versions of the same page. In one case, both pages contain similar number of links and the other is that both pages have quite different number of links.

For the first situation, examples found include:

- There are the same number of links within the page sent to the crawler and web browser, but the corresponding link pairs have a different format. For example, the link to web browser may contain a PHP session id while the link to the crawler does not. Another example is that the page to the crawler only contains absolute URLs, while the page to the browser contains relative URLs that are in fact pointing to the same targets as the absolute ones.
- The links in the page to the crawler are direct links, while the corresponding links within the page to web browser are encoded redirections.
- The links to web browser are normal links, but the links to the crawler are around small images instead of texts.
- The website shows links to different style sheets to web browser than to the crawler. For example, the page to the crawler contains "href=/styles/styles_win_ie.css", while the page to the browser contains "href=/styles/styles_win_ns.css".

In some cases, the number of links within the page to the crawler and the page to the web browser can be quite different.

- More links exist in the page sent to the crawler than the page sent to web browser. For example, these links may point to a link farm.
- The page sent to web browser has more links than the page sent to the crawler. For example, these links may be navigational links.
- The page sent to the browser contains some normal links, but in the same position of the page sent to the crawler, only error messages saying "no permission to include links" exist.

From the results shown within this section, it is obvious that cloaking is not rare in the real Web. It happens more often for hot queries or popular topics.

5 Results of Redirect

As we have discussed in Section 1, redirection can also be used as a spamming technique. To get an insight into how often the redirection appear and distribution of different redirect methods, we use the HITSdata set mentioned in Section 3. We don't use all four copies but only compare two copies for each page: one from the simulated browser's set (*BROWSER*) and the other from the crawler's set (*CRAWLER*).

5.1 Distribution

We check the distribution of four different types of redirection: HTTP 301 Moved Permanently and 302 Moved Temporarily responses, the HTML meta refresh tag, and the use of JavaScript to load a new page.

In order to know the distribution of above four different redirects, we tabulated the number of appearances of each type. For the first two types, the situation is simple: we just count the pages with response status of "301" and "302". The last two are more complicated; the HTTP refresh tag does not necessarily mean a redirection and JavaScript

TYPE	CRAWLER	BROWSER
301	20	22
302	56	60
Refresh tag	4230	4356
JavaScript	2399	2469

Table 7: Number of pages using different types of redirection.

is even more complicated for redirection purpose. For the first step, we just count the appearance of "<meta http-equiv=refresh>" tag for the third type and count the appearance of "location.replace" and "window.location" for the fourth type. The results for this first step are shown in Table 7.

To get a more accurate number of appearances of the HTTP refresh tag, we examined this further. In reality, the *Refresh* tag may just mean refreshing, not necessarily to redirection to another page. For example, the *Refresh* tag may be put inside a *NOSCRIPT* tag for browsers that do not support JavaScript. To estimate the number of real redirection by using this refresh tag, we randomly selected 20 pages from the 4,230 pages that use the refresh tag and checked them manually. We found that 95% of them are real redirection and only 5% are inside a *NOSCRIPT* tag. Besides, some pages may have identical target URL as themselves in the *Refresh* tag to keep refreshing themselves. We also counted these numbers. There are 47 pages out of 4,230 pages within the *CRAWLER* data set and 142 pages out of 4,356 pages within the *BROWSER* data set that refresh to themselves.

We did one more step for the 4,214 (4,356 - 142) pages that are pages using *Refresh* tag and refresh to a different page. Usually there is a time value assigned within the refresh tag to show how long to wait before refreshing. If this time is small enough, i.e., 0 or 1 seconds, users can not see the origin page but are redirected to a new page immediately. We fetched this time value for these 4,214 pages and draw the distribution of different time values from the range of 0 seconds to 30 seconds in Figure 9. More than 50% of these pages refresh to a different URL after 0 seconds and about 10% refresh after 1 second.

To estimate the real distribution of the JavaScript refresh method, we randomly selected 40 pages from the 2,399 pages that have been identified as candidates for using JavaScript for redirection in the first step. After manually checking these 40 files, we found the 20% of them are real redirections, 32.5% of them are conditional redirections, and the rest are not for redirection purpose, such as to avoid showing the

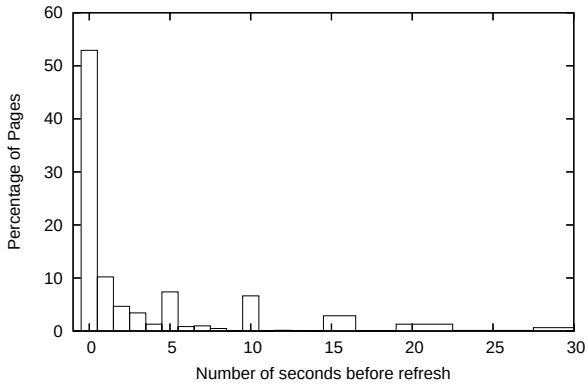


Figure 9: Distribution of delays before refresh.

page within a frame.

Sometimes the target URL and origin URL are within the same site, while other times they are on different sites. In order to know the percentage of redirections that redirection to the same sites, we also analyzed our collected data set for this information. Since the JavaScript redirection is complicated, we only count the first three types of redirection here. The sum of the first three types of redirection is 4,306. Within the *CRAWLER* data set, there are 2,328 pages within these 4,306 pages redirecting to the same site, while for the *BROWSER* data set, the number is 2,453.

5.2 Redirection Cloaking

As we have mentioned in Section 4.3, the site may return pages redirecting to different locations in case of different user agents. We consider this redirection cloaking.

We found that there are 153 pairs of pages out of 250,000 pairs that have different response code for a crawler and a normal browser when doing redirecting. Usually these web sites will send 404 or 503 response code to one and send 200 response code to the other. We even found that there are 10 pages that use different redirection method for a crawler and normal web browser. For example, they may use 302 or 301 for the crawler, but use refresh tag with the response code 200 for a normal web browser.

6 Summary and Discussion

Detection of search engine spam is a challenging research area. Cloaking and redirection are two impor-

tant spamming techniques.

This study is based on a sample of a quarter of million pages and top responses from a popular search engine to hot queries on the Web. We identified different kinds of cloaking and gave an estimate of the percentage of pages that are cloaked in the sample and also show an estimation of distribution of different redirect.

There are four issues that we would like to see addressed in future work. The first is that of a bias in the dataset used. Our data sets (pages in or near the top results for queries) do not nearly reflect the Web as a whole. However, it might be argued that it reflects the Web that is important (at least for the purposes of finding pages that might affect search engine rankings through cloaking). The second is that this paper does not address IP-based cloaking, so there are likely pages that do indeed provide cloaked content to the major engines when they recognize the crawling IP. We would welcome the partnership of a search engine to collaborate on future crawls.

The final issue is the bottom line. While search engines may be interested in finding and eliminating instances of cloaking, our proposed technique requires three or four crawls. Ideally, a future technique would incorporate a two-stage approach that identifies a subset of the full web that is more likely to contain cloaked pages, so that a full crawl using a browser identity would not be necessary.

Our hope is that this study can provide a realistic view of the use of these two techniques and will contribute to robust and effective solutions to the identification of search engine spam.

References

[1] AskJeeves / Teoma Site Submit managed by ineedhits.com: Program Terms, 2005. Online at http://ask.ineedhits.com/programterms.asp.

[2] America Online, Inc. AOL Search: Hot searches, Mar. 2005. http://hot.aol.com/hot/hot.

[3] Ask Jeeves, Inc. Ask Jeeves About, Mar. 2005. http://sp.ask.com/docs/about/jeevesiq.html.

[4] M. Cafarella and D. Cutting. Building Nutch: Open source. *ACM QUEUE*, 2, Apr. 2004.

[5] Google, Inc. Google information for webmasters, 2005. Online at http://www.google.com/webmasters/faq.html.

[6] Google, Inc. Google Zeitgeist, Jan. 2005. http://www.google.com/press/zeitgeist/zeitgeist-jan05.html.

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

[8] M. R. Henzinger, R. Motwani, and C. Silverstein. Challenges in web search engines. *SIGIR Forum*, 36(2), Fall 2002.

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

[10] Lycos. Lycos 50 with Dean: 2004 web's most wanted, Dec. 2004. http://50.lycos.com/121504.asp.

[11] C. D. Manning and H. Schutze. *Foundations of statistical natural language processing*, chapter 8, pages 268–269. MIT press, 2001.

[12] M. Najork. System and method for identifying cloaked web servers, Jul 10 2003. Patent Application number 20030131048.

[13] A. Perkins. White paper: The classification of search engine spam, Sept. 2001. Online at http://www.silverdisc.co.uk/articles/spam-classification/.

[14] WebmasterWorld.com. Cloaking, 2005. Online at http://www.webmasterworld.com/forum24/.

[15] B. Wu and B. D. Davison. Identifying link farm spam pages. In *Proceedings of the 14th International World Wide Web Conference, Industrial Track*, May 2005.

[16] Yahoo! Inc. Yahoo! Help - Yahoo! Search, 2005. Online at http://help.yahoo.com/help/us/ysearch/deletions/.

Pagerank Increase under Different Collusion Topologies

Ricardo Baeza-Yates
ICREA-Univ.Pompeu Fabra
and University of Chile
ricardo.baeza@upf.edu

Carlos Castillo
Dept. of Technology
Universitat Pompeu Fabra
carlos.castillo@upf.edu

Vicente López
Cátedra Telefónica
Universitat Pompeu Fabra
vicente.lopez@upf.edu

Abstract

We study the impact of collusion –nepotistic linking– in a Web graph in terms of Pagerank. We prove a bound on the Pagerank increase that depends both on the reset probability of the random walk ε and on the original Pagerank of the colluding set. In particular, due to the power law distribution of Pagerank, we show that highly-ranked Web sites do not benefit that much from collusion.

1 Introduction

This paper studies the effects of different linking topologies in the ranking function induced by the Pagerank algorithm [13]. The Pagerank algorithm receives as input an adjacency matrix $L_{N\times N}$, where N is the number of Web pages, and renormalizes each row of L to sum 1, generating a transition matrix A. This transition matrix is slightly modified by adding a "random jump", i.e.: a transition from each node to each of the other nodes using the uniform transition matrix – a matrix U such that $u_{ij} = 1/N$.

$$P = (1-\varepsilon)A + \varepsilon U \tag{1}$$

The Pagerank algorithm calculates the probabilities p_i of the stationary state of the Markovian process induced by matrix P. That is, the eigenvector x corresponding to the largest eigenvalue (which in the case of this matrix is $\lambda_1 = 1$) of the matrix P:

$$P^T x = x$$

The Pagerank value of a page is used as an estimator of the quality of a Web page based on properties of the Web graph. The rationale for this estimation is that a high quality page is a page with many in-links coming from other high-quality pages.

This algorithm is expected to work much better than simply counting in-links, as it might be more resistant to what is called a "Sybil attack" [5]. A Sybil attack is an attempt of altering a recommendation system by creating multiple identities; in this context, this means creating multiple pages pointing to a single page.

The resistance of Pagerank to a Sybil attack comes from the fact that the pages created for the attack can only inherit the reputation they are currently receiving. However, in the case of Pagerank, the minimum is not zero, as even without in-links a page gets a minimum score of $\frac{\varepsilon}{N}$.

The strategy of creating many pages pointing to a single page is actually used on the Web, in fact, currently there are thousands or millions of Web pages created specifically for the objective of deceiving the ranking function of search engines [6]. "Because the Web environment contains profit seeking ventures, attention getting strategies evolve in response to search engine algorithms. For this reason, any evaluation strategy which counts replicable features of web pages is prone to manipulation"[13].

To the best of our knowledge, Pagerank by itself is not used as the sole indicator of quality by any of the larger search engines, but it is still an important part of the ranking function of some of them.

In this paper:

- We present an analysis for collusion under a more general case than the one presented in [15], this is, we consider the original links that the colluding set has.
- We prove that for a single page there is always something to win by colluding with other pages.
- We prove that the expected returns from collusion are lower for highly-ranked pages.

The rest of this paper is organized as follows: Section 2 summarizes previous work in this area, and Section 3 presents an analysis predicting the increase of Pagerank by using a collusion strategy. Section 4 validates these predictions in a synthetic graph, and Section 5 studies a real Web graph. Finally, Section 6 presents our conclusions and avenues for future work.

2 Previous Work

Several authors have observed the presence of spam pages on the Web. Fetterly *et al.* [7] showed that most of the outliers when observing statistics of Web page collections are machine-generated spam pages –these pages may be designed both to increase citation counts and to provide multiple "doorway" pages. Hence, the divergence between the expected and the observed values can be used in some cases to detect spam pages.

Eiron *et al.* [6] studied a 100-million page sample and found that 11 of the top 20 URLs by Pagerank were pornographic, and in all cases the specific technique used was taking the Pagerank from random teleportation in many pages and concentrate it into a single page by using links.

Zhang *et al.*[15] study the following collusion strategy: pick a series of nodes with adjacent rankings, remove all their out links and add links from each node to the node before and after it in the list of nodes sorted by Pagerank. They also prove the following upper bound on any collusion strategy; let x_{orig} be the original Pagerank of a page, and x_{new} the Pagerank this page obtains after it colludes is:

$$\frac{x_{new}}{x_{orig}} < \frac{2}{\varepsilon}$$

They also prove that this bound is near $1/\varepsilon$ if $M \ll N$, as a typical value for ε is 0.15, the amplification factor is roughly 7. They do not take into account the starting Pagerank of the colluding set. We prove a tighter bound that shows that colluding works mostly for pages with low starting Pagerank.

Meyer [11] proved that if the second eigenvalue of an irreducible Markov chain is small, then the chain is not overly sensitive to small variations. Haveliwala and Kamvar [9] proved that the second eigenvalue of P is $1-\varepsilon$, therefore, a large ε produces a more stable matrix. Ng *et al.* [12] prove a similar result using a different approach: as long as ε is not too small, small variations in the matrix do not generate large variations of Pagerank.

Agogino and Ghosh [1] studied a reinforcement learning method for automatically finding a linking strategy for increasing the combined Pagerank of a set of domains. Their strategy relies on a utility function that considers the impact of every learner in the total Pagerank achieved by the colluding group.

Clausen [3] studied the cost of an attack on Pagerank considering that creating a new Web site requires a payment. In the same paper there is an analysis on how to lump pages together for Pagerank calculation, disregarding the internal link structure of each group.

In a recent article, Gyöngyi and Garcia-Molina [8] study optimal structures for "link spam farms" and combinations of them. A spam farm is an arrangement of links with the objective of increasing the ranking of a single target page. They prove that the optimal structure for a spam farm is a series of pages pointing to and only to the target page, while the target page points to some of all of them. In this optimal structure, if there are other external, "hijacked" links, they should also point to the target page.

3 Impact of Collusion in Pagerank

A group of nodes can collude to get a higher Pagerank by manipulating the out-links of the group. We will assume that the group's objective is to maximize its *total* Pagerank value.

Let N be the total number of nodes in Web graph G, M the number of colluding nodes in sub-graph G' – we will assume $M \ll N$. Let x be the total Pagerank of the colluding nodes, so $1-x$ is the total Pagerank of the rest of the graph. All the links in this graph are shown in Figure 1.

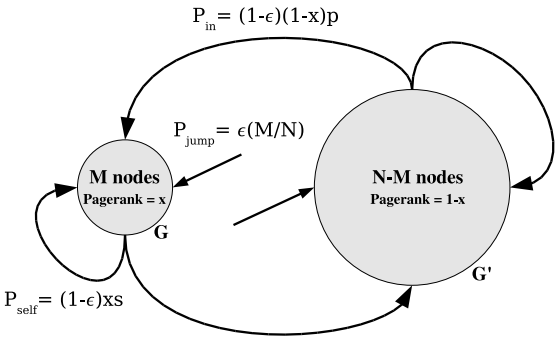


Figure 1: Variables used in the analysis.

The total Pagerank entering the colluding nodes PR_{in} is given by the sum of three terms representing links from random jumps, links from the non-colluding nodes and internal links between colluding nodes. This is the same approach taken by Clausen [3] to "lump" a set of pages into a single node for Pagerank computation.

$$Pagerank_{in} = P_{jump} + P_{in} + P_{self}$$

For calculating P_{in}, we first take the sum over all nodes pointing to the colluding set, instead of on all the links:

$$\begin{aligned} P_{in} &= \sum_{(a,b):(a,b)\in E_{in}} \frac{PR_a}{deg(a)} \\ &= \sum_{a:a\in G-G'} PR_a p(a) \end{aligned}$$

Where $p(a)$ is the fraction of links from node a pointing to the colluding set G', and it can be zero if no link from node a points to G'.

Now, let:

$$\begin{aligned} p &= \frac{\sum_{a:a\in G-G'} PR(a)p(a)}{\sum_{a:a\in G-G'} PR(a)} \\ &= \frac{\sum_{a:a\in G-G'} PR(a)p(a)}{1-x} \end{aligned}$$

So p is a weighted average of $p(a)$ over $G-G'$, in which the weights are the Pagerank values of the nodes in $G-G'$. The important issue is that p cannot be controlled by the colluding nodes, and will remain constant whatever strategy is used. We can now write the equation for P_{in} as:

$$P_{in} = (1-\varepsilon)(1-x)p$$

For P_{self} we make a similar replacement. If $s(b)$ is the fraction of links from node $b \in G'$ pointing to the colluding set G', then let:

$$\begin{aligned} s &= \frac{\sum_{b:b\in G'} PR(b)s(b)}{\sum_{a:a\in G'} PR(b)} \\ &= \frac{\sum_{b:b\in G'} PR(b)s(b)}{x} \end{aligned}$$

So s represents a weighted average of $s(a)$ over G', and this yields:

$$P_{self} = (1-\varepsilon)xs$$

Now we can write the equation for the sum of the Pagerank of the colluding nodes as:

$$Pagerank_{in} = \varepsilon\frac{M}{N} + (1-\varepsilon)(1-x)p + (1-\varepsilon)xs$$

Solving the stationary state $Pagerank_{in} = x$ yields:

$$x_{orig} = \frac{\frac{\varepsilon M}{N} + (1-\varepsilon)\,p}{(p-s)(1-\varepsilon)+1}$$

The only thing the colluding nodes can do is to link more internally than externally. This means that $s \to s'$, with $s' > s$, and the ratio between the resulting Pagerank and the original Pagerank is:

$$\frac{x_{new}}{x_{orig}} = 1 + \frac{s'-s}{p-s'+\frac{1}{1-\varepsilon}} \tag{2}$$

A trivial observation is that if $s' > s$ then:

$$\frac{x_{new}}{x_{old}} > 1$$

That is, there is always something to win by colluding with other nodes. In particular, colluding by forming a clique means $s' \to 1$, and the ratio between the resulting Pagerank x_{new} and the original Pagerank is:

$$\frac{x_{new}}{x_{old}} = 1 + \frac{1-s}{p+\frac{\varepsilon}{1-\varepsilon}} \tag{3}$$

This ratio is inversely proportional to the Pagerank that originally entered the colluding nodes. Therefore, if the colluding set has a high connectivity at the beginning, the returns from colluding will be poor, and viceversa.

For instance, if the starting set has very few, or no in-links from the rest of the graph, $p = 0$, and at the beginning $s = 0$ (originally all the out-links went to the rest of the graph), then:

$$\frac{x_{new}}{x_{old}} = \frac{1}{\varepsilon} \approx 7$$

If the starting Pagerank is very good, and has all the links from the non-colluding set we have $p = 1$, but has no internal links at the beginning, then:

$$\frac{x_{new}}{x_{old}} = 2-\varepsilon$$

However, this situation is very unlikely, because if the colluding set has all the in-links from the rest of the graph, then it should also have some links from itself. In fact, if we assume that in the original situation, the fraction of links going to the colluding nodes was the same for all nodes in the graph, then $s = p$, and the change in Pagerank value is given by the following equation:

$$\frac{x_{new}}{x_{old}} = \frac{1}{p(1-\varepsilon)+\varepsilon} \tag{4}$$

The graph of Pagerank change for $\varepsilon = 0.15$ and varying values of p is shown in Figure 2. We can see that while the starting fraction of links received remains roughly below 1% the returns are still maximum.

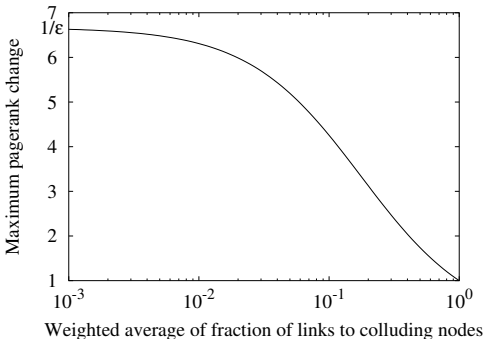


Figure 2: Expected change of Pagerank values under different starting conditions p, using $\varepsilon = 0.15$.

Focus on a single page In this section we have discussed the issue of increasing the *average Pagerank* of a set of pages. This is not the same as increasing the Pagerank of a single page as in the link spam farm structures studied in [8]. A simple, brute-force strategy of creating M (unlinked) pages, each with a single link pointing to a target page yields a Pagerank for the latter of at most:

$$\begin{aligned} x_{brute-force(M)} &= \frac{\varepsilon}{N} + (1-\varepsilon)\varepsilon\frac{M}{N} \\ &= \frac{\varepsilon + \varepsilon M - \varepsilon^2 M}{N} \\ &\approx \varepsilon\frac{M}{N} \qquad (M \gg 1; \varepsilon \ll 1) \end{aligned}$$

This is, an individual page can get a increase of Pagerank larger than in our analysis, but the average amplification of all the pages in the colluding set will be as described by Equation 4.

4 Experiments with a Synthetic Web Graph

We obtained a synthetic graph using a generative model described by Kumar *et al.* [10]:

- Nodes are added one at a time.
- Each time a node is added, d links are added. For adding a link, the source and destination nodes are chosen as follows:
  - With probability β the source node is chosen at random, and with probability $1-\beta$ the source node is chosen with probability proportional to the current out-degree of nodes.
  - With probability α the destination is chosen at random, and with probability $1-\alpha$ the destination is chosen with a probability proportional to the current in-degree of nodes.

We used $d = 7$, $\alpha = 0.2$ and $\beta = 0.45$, parameters experimentally determined by Pandurangan *et al.* [14] that produce graphs simultaneously fitting the distributions of in-degree, out-degree and Pagerank to the values observed in real Web graphs. The parameters for the power-law in the center part of the distributions are -2.1 for in-degree and Pagerank, and -2.7 for out-degree.

It is very important to remove the disconnected nodes from the resulting graph, as they affect the Pagerank normalization factor. This is specially critical for groups of pages with very low starting ranking, as they will certainly include the disconnected nodes, and those nodes will become connected after the collusion, modifying the number of nodes involved in the total Pagerank calculation. Using the generative model described above, we created a 125,000-nodes graph and then removed all the disconnected nodes to obtain a connected graph of roughly 106,000 nodes. We also made preliminary experiments with a 10,000-nodes graph and the results were very similar.

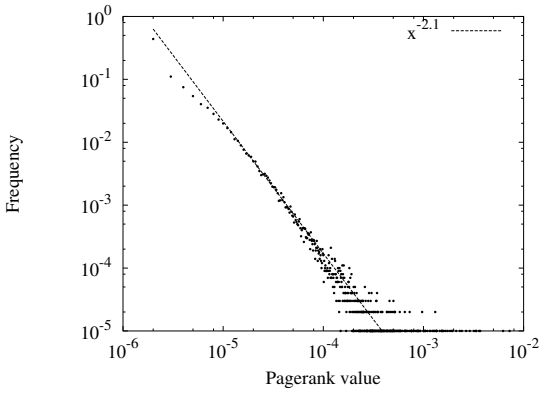


Figure 3: Distribution of Pagerank values in the synthetic graph.

Instead of sampling according to the number of nodes, we sampled according to the amount of Pagerank. We divided the nodes in the Web graph into 10 segments, each segment having $1/10^{th}$ of the total Pagerank. Note that because of the distribution of Pagerank, shown in Figure 3, these segments represent sets exponentially decreasing in size.

Inside each segment, we picked a group of $M = 100$ nodes at random –except in the last segment, as the top 10% of Pagerank was found in only 50 nodes. We labeled these groups $1 \ldots 10$.

In the following, we denote by **Pagerank values** the actual probabilities given by the Pagerank algorithm, this is, the resulting values of the vector x. We denote by **ranking** the order in which a page appears when pages are sorted by Pagerank values. This number is normalized so 0 is the last page and 1 is the top page by Pagerank value.

The original Pagerank values of the pages inside each group, as well as the group averages, are shown in Figure 4.

As the distribution of the Pagerank values is very skewed, the distribution of the rankings inside each of these groups appears as shown in Figure 4. Note that most of the pages in group 1 have the same Pagerank value, so their ranking is distributed uniformly in the 0.0-0.5 interval, meaning that the bottom 50% of the pages have in total just 10% of the Pagerank.

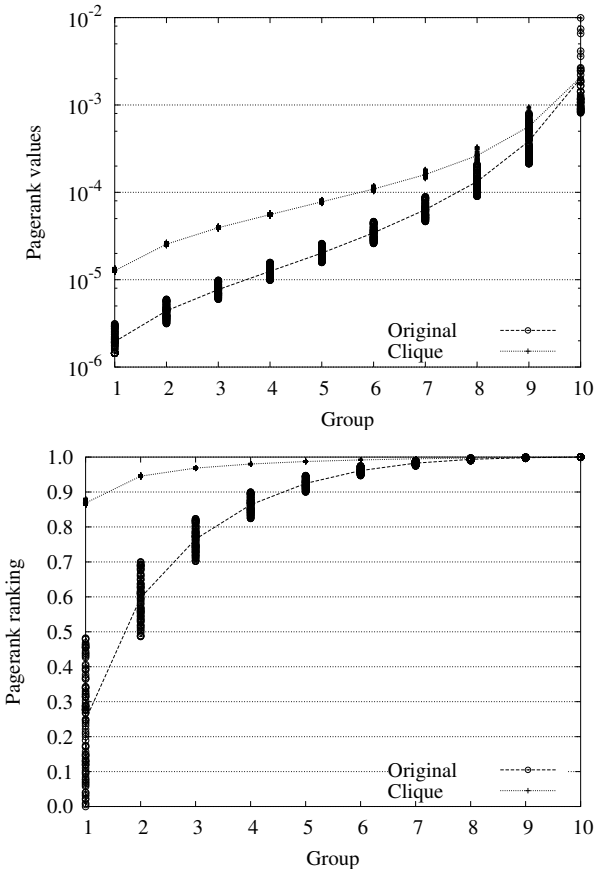


Figure 4: Pagerank values and rankings, in both the original and the modified graphs, using a clique inside each group.

4.1 Collusion via a complete sub-graph

The first strategy we tested was to create a clique (a complete sub-graph) inside each group. Unlike the experiments by [15], we did not remove any outgoing links, as that is very easy to detect and can be penalized by search engines. The rationale is that if within a group the number of internal links outnumbers the number of external links then that group will preserve its Pagerank.

Figure 4 compares the Pagerank values before and after the collusion.

Figure 5 plots the variation in both Pagerank values and ranking after collusion. Clearly the bound $1/\varepsilon$ is too coarse for medium to highly ranked pages, in those cases, the starting incoming links should be considered, as in Equation 4.

Note that a factor of 3 in the Pagerank ranking is a very large increase for a page. As shown in Figure 5, all of the pages from group one get a new ranking in the top 10% by colluding. Moreover, all of the pages inside each group get roughly the same ranking.

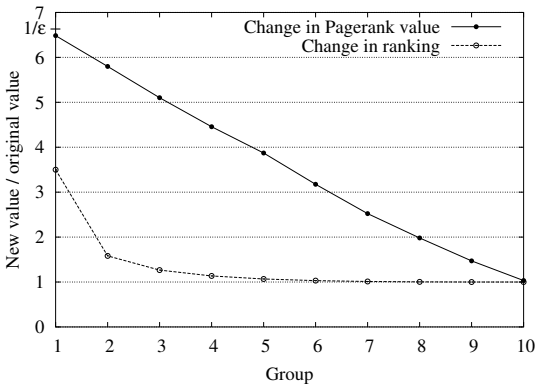


Figure 5: Relative variation of Pagerank value and ranking using a clique inside each group. Link spamming only yields returns for pages with low starting Pagerank values.

4.2 Collusion via a partial sub-graph

It might not be necessary to create all links, just enough links to keep most of the Pagerank inside the colluding set. We tested varying amounts of linking in the synthetic graph. The amplification factor obtained by colluding is shown in Figure 6.

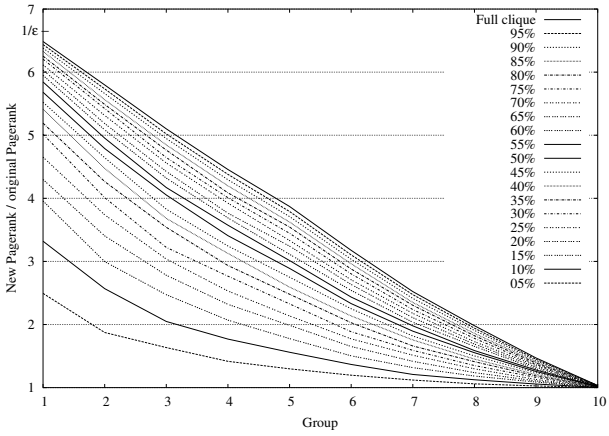


Figure 6: Pagerank change under varying amounts of internal links.

In most cases, adding just 50% of the links yields high returns, and in the case of the group with the lower starting Pagerank, even 30% of the links results in an increase of Pagerank by a factor of 5.

4.3 Other collusion strategies

We also tested collusion strategies involving $O(M)$ internal links instead of $O(M^2)$ as is the case with cliques. The two topologies we studied were a star and a ring, as depicted in Figure 7.

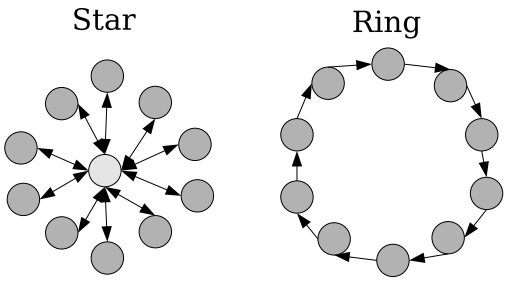


Figure 7: Studied linking topologies.

In the case of the star, to avoid a positive bias in the choice of the center of the star, we picked a new node originally without in-links as the center of the star. The comparison between the results under these two topologies is shown in Figure 8. There is a slight advantage of forming a star instead of a ring for the lowly-ranked sites, but for the other groups both strategies yield similar returns, and those are much lower than in the case of cliques.

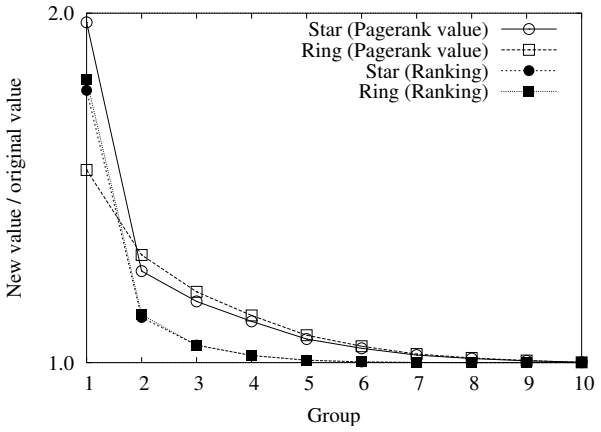


Figure 8: Pagerank under other linking topologies; both the star and ring topologies yield much lower returns than forming a clique.

5 Experiments with a Real Web Graph

We started with a collection of 16 million pages from Spanish Web sites obtained during 2004. In this case, we are interested in complete Web sites instead of individual pages, so we first converted multiple links between pages in different Web sites, into a single link between two Web sites. Two sites s_1, s_2 are linked iff there is at least one page on site s_1 pointing to a page in s_2.

We obtained a graph with $310,486$ Spanish sites and $3,037,913$ directed links between them.

We calculated the Pagerank (or Hostrank [4]) values of each Web site, and the corresponding ranking induced by this value. Note that this is not the same as the sum of the page-wise Pagerank for each page in the Web site, because there may be multiple links between two Web sites [2]. The distribution of values for the Hostrank is shown in Figure 9.

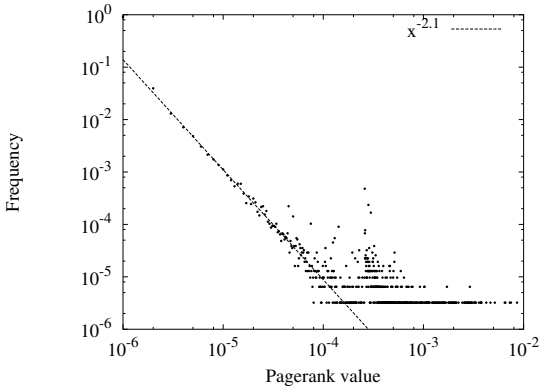


Figure 9: Distribution of Hostrank values in the Spanish Web sites graph. There are a number of Web sites already colluding.

Comparing this with the distribution of Pagerank in the synthetic graph, shown in Figure 3, we can see that while both exhibit a Zipf's law with roughly the same parameter, in the real Web there is a significant number of outliers. Manual inspection of these outliers showed that most of them are Web sites that can be considered as spam, for instance, we found several groups of dozens of Web sites with names such as `http://cityname.company.es/`, in which `cityname` is the name of a Spanish city and `company` is a tour operator or hotel company.

We modified this graph with the strategies we have discussed so far and computed the new Pagerank values after each strategy. The objective of these strategies is to increase the ranking of a small set of 242 sites (0.08% of the total number of sites). The target sites for this experiments were obtained from the directory of an agency certifying

the quality of Spanish Web sites, and are expected to be sites that adhere to certain standards of coding, content, etc.

The average ranking of the Web sites in the selected group is 0.75. Note that this only takes into account the Pagerank value, while the quality of a site may come from very different factors.

Table 1: Linking strategies.

Strategy	Average ranking
Disconnect group	0.75
Normal	0.77
Central site	0.82
Ring, alphabetical	0.93
Ring, inverted	0.93
Star	0.96
Clique	0.99

Table 1 lists the linking strategies used. We started by disconnecting all the links between the participating nodes, that yields a minimum of ranking without collusion at all. After that, we returned to the normal situation. Then we added a central site that lists all the participating sites, then a ring of all the sites in alphabetical order, and then an inverted ring. Finally, we also added a star and a clique. Figure 10 lists the resulting rankings under different strategies.

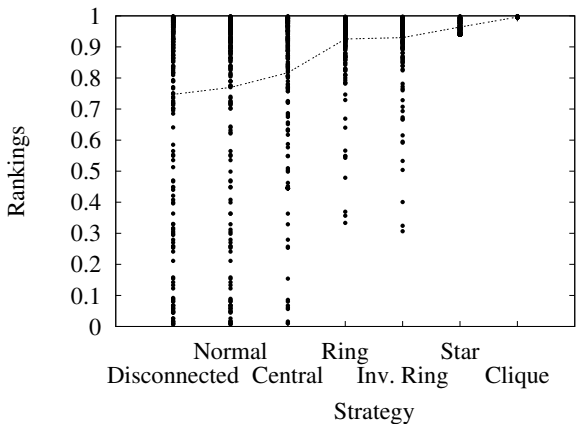


Figure 10: Relative rankings under different strategies. Each dot represents a site.

Clearly, creating a complete subgraph is the best strategy, as all of the sites in the group get very high positions. We noted that the star strategy gets a similar average to the ring strategy, but much lower variability.

Finally, we explored the possibility of adding less than 50% of the links of a complete subgraph. In Figure 11, a varying amount between 5% and 50% of the links in the complete subgraph are added randomly.

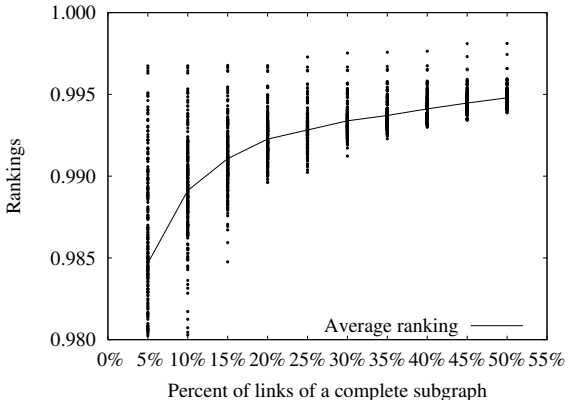


Figure 11: Relative rankings when adding a fraction of the links of a complete subgraph.

We observe that even adding 5% of the links of a complete subgraph, i.e.: each of the Web sites in the group links to 5% of the other sites (in this case, roughly 10 Web sites each one), then the average position (0.985) is higher than all the strategies based on other topologies. After adding about 20% of the links of the complete subgraph, the gains increase linearly with the number of links.

6 Conclusions and Future Work

While any group of nodes can increase their Pagerank by forming a tightly-connected sub-graph of the Web, the increase they obtain by doing so is inversely related to their starting Pagerank. This means that the Pagerank algorithm is particulary vulnerable to Sybil attacks from the nodes with low Pagerank. As the distribution of Pagerank is very skewed, even a modest increase in Pagerank value may imply a large increase in the ranking of a page.

Collusion strategies have never been studied in a more microscopic scale. For instance, we noted slight differences when creating a ring of pages in two different orderings. There is an optimum ordering for forming a ring, and we are interested in studying different strategies under a limited “budget” in terms of links.

As future work, we would like to study other forms of ranking calculation such as a two-level ranking scheme that ranks entire Web sites and then Web pages.

While this article is mainly descriptive, we are also interested in developing ways of detecting deceptive linking practices to improve reputation algorithms. There is not

a trivial answer to this problem. Finding regular structures [7] may not be enough as spammers can randomize their link spam farms. Measuring the ratio between the total Pagerank of a group of pages and the Pagerank they receive externaly [15] may detect groups of pages that are strongly linked among them for legitimate reasons. An open question is if the current linking practices used amongst "good sites" should be used –and accepted– or not.

References

[1] AGOGINO, A., AND GHOSH, J. Increasing pagerank through reinforcement learning. In *Proceedings of Intelligent Engineering Systems Through Artificial Neural Networks* (St. Louis, Missouri, USA, November 2002), vol. 12, ASME Press, pp. 27–32.

[2] BAEZA-YATES, R., AND CASTILLO, C. Relating Web characteristics with link based Web page ranking. In *Proceedings of String Processing and Information Retrieval SPIRE* (Laguna San Rafael, Chile, November 2001), IEEE CS Press, pp. 21–32.

[3] CLAUSEN, A. The cost of attack of PageRank. In *Proceedings of the international conference on agents, Web technologies and Internet commerce (IAWTIC)* (Gold Coast, Australia, July 2004).

[4] DILL, S., KUMAR, R., MCCURLEY, K. S., RAJAGOPALAN, S., SIVAKUMAR, D., AND TOMKINS, A. Self-similarity in the web. *ACM Trans. Inter. Tech. 2*, 3 (2002), 205–223.

[5] DOUCEUR, J. The Sybil Attack. In *Proceedings of the first International Peer To Peer Systems Workshop (IPTPS)* (Cambridge, MA, USA, January 2002), vol. 2429 of *Lecture Notes in Computer Science*, Springer, pp. 251–260.

[6] EIRON, N., MCCURLEY, K. S., AND TOMLIN, J. A. Ranking the web frontier. In *Proceedings of the 13th international conference on World Wide Web* (New York, NY, USA, 2004), ACM Press, pp. 309–318.

[7] FETTERLY, D., MANASSE, M., AND NAJORK, M. 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, June 2004).

[8] GYÖNGYI, Z., AND GARCIA-MOLINA, H. Link spam alliances. Tech. rep., Stanford University, California, March 2005.

[9] HAVELIWALA, T., AND KAMVAR, S. The second eigenvalue of the Google matrix. Tech. rep., Stanford University, 2003.

[10] KUMAR, R., RAGHAVAN, P., RAJAGOPALAN, S., SIVAKUMAR, D., TOMKINS, A., AND UPFAL, E. Stochastic models for the web graph. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS)* (Redondo Beach, CA, USA, 2000), IEEE CS Press, pp. 57–65.

[11] MEYER, C. D. Sensitivity of the stationary distribution of a Markov chain. *SIAM Journal on Matrix Analysis and Applications 15*, 3 (1994), 715–728.

[12] NG, A. Y., ZHENG, A. X., AND JORDAN, M. I. Link analysis, eigenvectors and stability. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)* (Seattle, Washington, USA, 2001), pp. 903–910.

[13] PAGE, L., BRIN, S., MOTWANI, R., AND WINOGRAD, T. The Pagerank citation algorithm: bringing order to the web. Tech. rep., Stanford Digital Library Technologies Project, 1998.

[14] PANDURANGAN, G., RAGHAVAN, P., AND UPFAL, E. Using Pagerank to characterize Web structure. In *Proceedings of the 8th Annual International Computing and Combinatorics Conference (COCOON)* (Singapore, August 2002), vol. 2387 of *Lecture Notes in Computer Science*, Springer, pp. 330–390.

[15] ZHANG, H., GOEL, A., GOVINDAN, R., MASON, K., AND ROY, B. V. Making eigenvector-based reputation systems robust to collusion. In *Proceedings of the third Workshop on Web Graphs (WAW)* (Rome, Italy, October 2004), vol. 3243 of *Lecture Notes in Computer Science*, Springer, pp. 92–104.

SpamRank – Fully Automatic Link Spam Detection*
Work in progress

András A. Benczúr[1,2] Károly Csalogány[1,2] Tamás Sarlós[1,2] Máté Uher[1]

[1] Computer and Automation Research Institute, Hungarian Academy of Sciences (MTA SZTAKI)
11 Lagymanyosi u., H–1111 Budapest, Hungary
[2] Eötvös University, Budapest
{benczur, cskaresz, stamas, umate}@ilab.sztaki.hu
www.ilab.sztaki.hu/websearch

Abstract

Spammers intend to increase the PageRank of certain spam pages by creating a large number of links pointing to them. We propose a novel method based on the concept of personalized PageRank that detects pages with an undeserved high PageRank value without the need of any kind of white or blacklists or other means of human intervention. We assume that spammed pages have a biased distribution of pages that contribute to the undeserved high PageRank value. We define SpamRank by penalizing pages that originate a suspicious PageRank share and personalizing PageRank on the penalties. Our method is tested on a 31 M page crawl of the `.de` domain with a manually classified 1000-page stratified random sample with bias towards large PageRank values.

1 Introduction

Identifying and preventing spam was cited as one of the top challenges in web search engines in a 2002 paper [24]. Amit Singhal, principal scientist of Google Inc. estimated that the search engine spam industry had a revenue potential of $4.5 billion in year 2004 if they had been able to completely fool all search engines on all commercially viable queries [36]. Due to the large and ever increasing financial gains resulting from high search engine ratings, it is no wonder that a significant amount of human and machine resources are devoted to artificially inflating the rankings of certain web pages.

In this paper we concentrate on identifying pages backlinked by a large amount of other pages in order to mislead search engines to rank their target higher. Our main goal is to compute for each Web page a SpamRank value that measures the amount of the undeserved PageRank [32] of a page. Note that by the nature of our methods we make no distinction between fair or malicious intent and our algorithm will likely rank pages with a large amount of low quality backlinks as spam.

In order to understand the nature of link spam, we first consider the characterization of an "honest" link by Chakrabati et al. [11]:

> "hyperlink structure contains an enormous amount of latent human annotation that can be extremely valuable for automatically inferring notions of authority."

*Support from NKFP-2/0017/2002 project Data Riddle and various ETIK, OTKA and AKP grants

Note that their notion of an authority plays similar role as a page with high PageRank value. In this sense the existence of a hyperlink should affect ranking only if it expresses human annotation. As examples for different uses of hyperlinks we refer to the article of Davison [13] that shows how to determine intra-site links that may serve both navigational and certain spam purposes. In [22] more examples are given, among others spamming guest books with links or mirroring with the sole purpose of the additional links to spam targets.

We believe that identifying email spam or web content spam by *human inspection* is relative easy and automated methods cannot, in any case, perform as good as human judgement. Györgyi and Garcia-Molina [22] list a few methods that confuse users including term hiding (background color text); cloaking (different content for browsers and search engine robots) and redirection; some of these techniques can still be found by inspecting the HTML code within the page source. Detecting redirection may already require certain expertise: we found quite a number of doorway spam pages which used obfuscated JavaScript code to redirect to their target.

Web link spam, in contrast, appears to be much harder to catch. As an example of a page that we ranked high for spam, 2way.handyfritz.de looks like an "innocent" site for mobile logos and tones while a large number of its backlinks are in fact spam messages in guestbooks. As a side effect we penalize pages often cited in blogs and lists, for example to www.golem.de/0501/35735.html. Certain forms of a link spam are however visible to Web users as well: the `thema-*.de` clique contains no useful content, only long list of links to itself and to various eBay.de auctions and pop-up ads fill its pages. The misuse of the Scout24.de affiliate program is also quite popular among the German spammers.

We also see frequent examples of non-spam with undeserved PageRank. For example for affiliate pages, link spamming is not always the purpose, but it is always a side-effect as they get paid to redirect traffic to certain sites. Since we make no analysis of content, we cannot adopt a more restrictive notion of spam favored by the search engine optimization community. Depending on the actual query where a page receives high PageRank, one may refer to techniques to attain the high rank as *boosting* if the page content is otherwise relevant to the query, see e.g. [22]. Our method, in this sense, could be called de-boosting with the purpose of assisting users to find pages where the maintainer has a lower budget on Search Engine Optimization (SEO).

Preferred notions of Web link or content spam could be the following. In [22] a content is called spam if it is completely irrelevant for the given search terms. However when implementing a search engine one aims to use ranking methods that select the most relevant highest quality content irrespective to the amount of search engine optimization for similar, but possibly lower quality content. In another definition [34] search engine spam consists of features that maintainers would not add to their sites if search engines didn't exist. Link spam however could act even without the presence of a search engine by misleading users to visit certain pages, in particular for the purpose of misusing affiliate programs.

An example of a site with quality content that we penalize high is city-map.de. Such sites are distributed to so many pages across a large number of domain names that attain undeserved PageRank for these pages. While methods such as HostRank [17] and the site boundary detection of Davison [13] also handle these sites, we also notice their undeserved PageRank value by giving penalties for these sites. The natural idea of combining methods is beyond the scope of the current report.

1.1 Our method: Where does your PageRank come from?

We assume that spammed pages have a biased distribution of *supporter* pages that contribute to the undeserved high PageRank value. As described in detail in Section 2, a node's PageRank is equal to the average of its personalized PageRank where personalization is over all Web pages. By the recent algorithm of [21]

Algorithm 1: Overall Structure of SpamRank Algorithm

for all Web pages i **do**
  Support$_{i,\cdot}$ $\leftarrow$ empty sparse vector of reals
Phase 1: Supporter Generation
generate nodes into vector Support$_{i,\cdot}$ that have high contribution in the PageRank of i
Phase 2: Penalization
for all Web pages i **do**
  give Penalty$_i$ based on irregular behavior of PageRank over Support$_{i,\cdot}$
Phase 3: SpamRank as Personalized PageRank (PPR) over Penalties
SpamRank $\leftarrow$ PPR(Penalty)

we are able to approximately compute all these values[1] in order to deduce a large fraction of the origin of the node's PageRank value.

PageRank distribution in your neighborhood: looks honest or spam? Our key assumption is that supporters of an honest page should not be overly dependent on one another, i.e. they should be spread across sources of different quality. Just as in the case of the entire Web, the PageRank distribution of an honest set of supporters should be power law. Particular examples that raise suspicion when a page receives its PageRank only from very low ranked pages (and then from a very large number of them); such a page has little quality support that makes the fairness of the large number of low-quality supporters questionable. Another example is a set of supporters, all with PageRank values falling into a narrow interval. In this case the large number of similar objects raise the suspicion that they appear by certain means of a cooperation.

The two key observations in detecting link farms, colluding pages or other means of PageRank boosting in the neighborhood of a page are the following:

- Portions of the Web are self-similar; an honest set of supporter pages arise by independent actions of individuals and organizations that build a structure with properties similar to the entire Web. In particular, the PageRank of the supporters follows a power law distribution just as the case for the entire Web.
- Link spammers have a limited budget; when boosting the PageRank of a target page, "unimportant" structures are not replicated.

A perfect form of a link spam is certainly a full replica of the entire Web that automatic link based methods are unable to distinguish from the original, honest copy. Our method hence targets at finding the missing statistical features of dishonest page sets. In our experiment the power law distribution acts as this feature; we remark that (i) other features may perform well and (ii) our algorithm can be fooled by targeting a link farm towards the particular statistical property we employ, hence in a practical application a large number of features should be combined that should probably include non-link based methods as well.

Our algorithm to define SpamRank, a value that measures the amount of undeserved PageRank score of a Web page, consist of three main steps; the overall structure is given in Algorithm 1. First we select the supporters of each given page by a Monte Carlo personalized PageRank simulation as described in Section 2. Then in the second phase we penalize pages that originate a suspicious PageRank share, i.e. their

[1] Approximate personalized PageRank values are stored space-efficiently in a sparse matrix

personalized PageRank is distributed with bias towards suspicious targets. This step is performed target by target; we measure the similarity of the PageRank histogram of sources to an ideal power law model suggested by [5, 27]. Then in the third step we simply personalize PageRank on the penalties. This last step concentrates suspicious activities to their targets; in another way to state, we determine SpamRank by a back-and-forth iteration between targets and sources.

1.2 Related work

With the advent of search engines web spamming appeared as early as 1996 [12, 2]. The first generation of search engines relied mostly on the classic vector space model of information retrieval. Thus web spam pioneers manipulated the content of web pages by stuffing it with keywords repeated several times.

Following Google's success all major search engines quickly incorporated link analysis algorithms such as HITS [26] and PageRank [32] into their ranking schemes. The birth of the highly successful PageRank algorithm [32] was indeed partially motivated by the easy spammability of the simple in-degree count. However Bianchini et al. [7] proved that for any link farm and any target set of pages such that each target page is pointed to by at least one link farm page the sum of PageRank over the target set's nodes is at least large as a linear function of the number of pages in the link farm.

Bharat and Henzinger [6] improved HITS to reduce its sensitivity to mutually reinforcing relationships between hosts. Generally, [31, 8, 35] discuss the (negative) effects of dense subgraphs (known as tightly-knit communities, TKCs) on HITS and other related algorithms.

Section 7 of [29] and the references therein give an overview of the theoretical results underlying the TKC effect that indicate a very weak TKC-type spam resistance of HITS and a somewhat better but still unsatisfying one of PageRank. The results show that HITS is unstable, its hub and authority ranking values can change by an arbitrary large amount if the input graph is perturbed. On the other hand, PageRank values are stable, but the ranking order induced by them is still unstable.

Davison [13] applied a decision tree trained on a broad set of features to distinguish navigational and link-spam (dubbed as nepotistic) links from the good ones. To the best of our knowledge Davison's work is the first openly published research paper *explicitly* devoted to the identification of link spam. More recently Amitay et al. [3] extracted features based on the linkage patterns of web sites. Clustering of the feature space produced a decent amount clusters whose members appeared to belong to the same spam ring.

Recently Fetterly et al. [18] demonstrated that a sizable portion of machine generated spam pages can be identified through statistical analysis. Outliers in the distribution of various web page properties – including host names and IP addresses, in- and out-degrees, page content and rate of change – are shown to be mostly caused by web spam.

Eiron et al. [17] gives evidence that HostRank – PageRank calculated over the host graph – is more resilient against link spam. HostRank's top list contained far fewer questionable URLs than PageRank's because of the relatively reduced weight given to link farm sites. This finding is in good agreement with the before mentioned linear spammability of PageRank [7].

Gyöngyi et al. [23] show that spam sites can be further pushed down in HostRank ordering if we personalize HostRank on a few trusted hub sites. Their method is semi automatic, the trusted 180 seed pages were carefully hand picked from 1250 good hub pages distilled automatically using Inverse PageRank[2]. From the same authors, [22] gives a detailed taxonomy of current web spamming techniques.

Zhang et al. [39] argue that the PageRank of colluding nodes (i.e. pages within the same dense, cliquish

[2] Although [23] makes no citation on Inverse PageRank, computing PageRank on the transposed web graph for finding hubs has been independently introduced previously in [14, 20].

subgraph) is highly correlated with $1/\epsilon$, where ϵ denotes the teleportation probability. Their method increases the probability of jump from nodes with large correlation coefficients. The resulting Adaptive Epsilon scheme is shown to be resistant against artificial, hand-made manipulations implanted by the authors to a real world web graph crawled by the Stanford WebBase project. Baeza-Yates et al. [4] improve Zhang et al.'s analysis and experimentally study the effect of various collusion topologies.

Wu and Davison [38] identify a seed set of link farm pages based on the observation that the in- and out-neighborhood of link farm pages tend to overlap. Then the seed set of bad pages is iteratively extended to other pages which link to many bad pages; finally the links between bad pages are dropped. Experiments show that a simple weighted indegree scheme on the modified graph yields significantly better precision for top ten page hit lists than the Bharat-Henzinger HITS variant. Moreover link farm sites with not too high original HostRank suffer a drastic loss when HostRank is calculated over the pruned graph.

As for a broader outlook, email spam is thoroughly discussed in the proceedings of the recently set up conference [16]. The sensitivity of e-commerce collaborative filtering algorithms to spam attacks is analyzed empirically in [28]. Dwork et al. [15] present spam resistant algorithms for rank aggregation in meta search engines.

2 Preliminaries

In this section we briefly introduce notation, and recall definitions and basic facts about PageRank. We also describe the Monte Carlo simulation for Personalized PageRank of [21].

Let us consider the web as a graph: let pages form a vertex set and hyperlinks define directed edges between them. Let A denote the stochastic matrix corresponding to random walk on this graph, i.e.

$$A_{ij} = \begin{cases} 1/\text{outdeg}(i) & \text{if page } i \text{ points to } j, \\ 0 & \text{otherwise.} \end{cases}$$

The *PageRank* vector $p = (p_1, \ldots, p_N)$ is defined as the solution of the following equation [9]

$$p_i = (1-\epsilon) \cdot \sum_{j=1}^{N} p_j A_{ji} + \epsilon \cdot r_i \,,$$

where $r = (r_1, \ldots, r_N)$ is the teleportation vector and ϵ is the teleportation probability. If r is uniform, i.e. $r_i = 1/N$ for all i, then p is the PageRank. For non-uniform r the solution p is called personalized PageRank; we denote it by $\text{PPR}(r)$. It is easy to see that $\text{PPR}(r)$ is linear in r [25]; in particular

$$p_i = \frac{1}{N} \sum_{v} \text{PPR}_i(\chi_v) \tag{1}$$

where χ_v is the teleportation vector consisting of all 0 except for node v where $\chi_v(v) = 1$.

By equation (1) we may say that the PageRank of page i arises as the contribution of personalization over certain pages v where $\text{PPR}_i(\chi_v)$ is high. We say that page v *supports* i to the above extent.

As noticed independently by [20, 25], the (personalized) PageRank of a vertex is equal to the probability of a random walk terminating at the given vertex where the length is from a geometric distribution: we terminate in step t with probability $\epsilon \cdot (1-\epsilon)^t$. To justify, notice that PageRank can be rewritten as a power series

$$\text{PPR}(r) = \epsilon \cdot \sum_{t=0}^{\infty} (1-\epsilon)^t r \cdot A^t. \tag{2}$$

The term $r \cdot A^t$ corresponds to a random walk of length t and $\epsilon \cdot (1-\epsilon)^t$ is the probability of termination.

The fact that PageRank can be computed as the probability over certain random walks gives rise to the following algorithm [21] that we use in our procedure. We generate large enough number of random walks starting at vertex j and add up probabilities $\epsilon(1-\epsilon)^t$ for their endpoints i; based on the counts we get $\text{Support}_{j,i}$ as an unbiased estimator of $\text{PPR}(\chi_j)_i$. Experiments in [21] suggest that a thousand simulations suffice in order to distinguish high and low ranked pages. The overall idea is summarized in Algorithm 2; we use the implementation described in [21] that differs from the above simplified description in two key aspects. First, for efficiency random walks are generated edge by edge; in one iteration each random walk is augmented by an edge. For such an iteration the set of random walks is sorted by their endvertex; then the iteration can be performed in a single edge scan of the Web graph. Finally we need a last sort over the set of random walks by endvertex; then for a single endvertex i all vertices are collected that support vertex i.

Algorithm 2: Phase 1 outline for finding supporters by Monte Carlo simulation. Actual implementation uses external sort [21].

for all Web pages i **do**
  for $\ell = 1, \ldots, 1000$ **do**
    $t \leftarrow$ random value from geometric distribution with parameter ϵ
    $j \leftarrow$ endvertex of a random walk of length t starting at i
    $\text{Support}_{j,i} \leftarrow \text{Support}_{j,i} + \epsilon(1-\epsilon)^t$

---

3 Algorithm

We define SpamRank, a measure of undeserved PageRank share of Web page, through a three-phase algorithm (Algorithm 1). The algorithm first identifies candidate sources of undeserved PageRank scores. In Phase 1 we select the supporters of each page by the Monte Carlo simulation of [21]. Then in Phase 2 pages receive penalties based on how many potential targets are affected and how strong is the influence on their PageRank values. Finally in Phase 3 we define *SpamRank* as PageRank personalized on the vector of penalties, in a similar way as, by folklore information, Google's BadRank [1] is computed (on the Web graph with reverse edge direction) personalized on identified spam.

In Phase 1 (Algorithm 2) we compute the approximate personalized PageRank vector of all pages j. We use the Monte Carlo approximation of [21]; this algorithm under practically useful parameter settings computes a set of roughly 1,000 nodes i together with a weight $\text{Support}_{i,j}$. This weight can be interpreted as the probability that a random PageRank walk starting at j will end in i.

Before proceeding with the penalty computation in Phase 2, we invert our data (by an external sort) and for each page i we consider the list of pages j such that i is ranked high when personalized on j; the strength is given by $\text{Support}_{i,j}$ as above. Notice that $\text{Support}_{i,j}$ arises from a Monte Carlo simulation and hence its value is 0 for all j where the actual personalized PageRank value is negligible.

For a fixed page i, penalties are defined by considering the PageRank histogram of all i with $\text{Support}_{i,j} > 0$ for pages that receive enough supporters. Pages with less than n_0 supporters (in our experiment $n_0 = 1000$) are ignored; supporter pages that spread their personalized PageRank to targets with less than n_0 incoming paths are of little spamming power anyway.

In the heart of our algorithm we find the method of identifying irregularities in the PageRank distribution of a page's supporters. Given such a measure $\rho \leq 1$ where $\rho = 1$ means perfect regularity, we proceed by

Algorithm 3: Phase 2 Penalty Calculation for Web pages, two variants.

Initialize vector Penalty by all 0
for all Web pages i with at least n_0 supporters j with nonzero $\text{Support}_{i,j}$ **do**
  $\rho \leftarrow$ regularity of the supporters of i
  if $\rho < \rho_0$ **then**
    for all Web pages j with $\text{Support}_{i,j} > 0$ **do**

$$\text{Penalty}_j \leftarrow \text{Penalty}_j + \begin{cases} (\rho_0 - \rho) & \{\text{Variant I}\} \\ (\rho_0 - \rho) \cdot \text{Support}_{i,j} & \{\text{Variant II}\} \end{cases}$$

    {we use $\rho_0 = 0.85$}
for all Web pages i **do**
  if $\text{Penalty}_i > 1$ **then**
    $\text{Penalty}_i \leftarrow 1$

penalizing all the supporter pages proportional to $(\rho_0 - \rho)$ if the measure is below a threshold ρ_0. In our experiments the variant where penalties are also proportional to the strength of the support, $\text{Support}_{i,j}$, proves slightly more effective. Also we put an upper limit of 1 for the penalties of a single page; penalties are otherwise accumulate for pages that participate in several irregular supporter sets.

3.1 Global properties of the Web graph

The fully automatic detection of irregular sets of supporters forms crucial part of our algorithm, hence we briefly describe the intuition behind our method. Key ingredients are

- Rich get richer evolving models: The in-degree and the PageRank of a broad enough set of pages should follow power law distribution.
- Self-similarity: A large-enough supporter set should behave similar to the entire Web.

We build on widely known models of the evolution of the Web [5, 27] that describe global properties such as the degree distribution or the appearance of communities. These models indicate that the overall hyperlink structure arises by copying links to pages depending on their existing popularity, an assumption agreeing with common sense. For example in the most powerful model [27] pages within similar topics copy their links that result in "rich gets richer" and we see a power law degree distribution.

The distribution of PageRank behaves very similar to that of the indegree as noticed among others in [33]. In all Web crawls considered by experiments PageRank has a power law distribution. Clearly PageRank and in-degree should be related as each page has its ϵ/N teleportation share of PageRank and propagates this value through out-links. Figures about PageRank and in-degree correlation vary; some claim a value close to 0 but typically a moderate value close to 0.5 is reported; as an example, the authors of [30] corrected their initial low correlation value to 0.34 in personal communication.

When looking at individual pages, models could in theory completely lose all their predictive power. In practice however strong self-similarity of various portions of the Web is observed [5] that may indicate that the PageRank in a neighborhood should have the same statistical properties as in the entire Web.

Stating in an opposite way, we argue that the neighborhood of a spam page will look different from an honest one. The neighborhood of a link spam will consist of a large number of artificially generated links. These links likely come from similar objects; the same fine granularity obtained by the rich gets richer principle is harder to be locally replicated.

3.2 Phase 2: Penalty generation

We are ready to fill in the last detail (Algorithm 4) of our SpamRank algorithm. Firstly for a page i we may consider the histogram of either the PageRank of all of its supporter pages j with $\text{Support}_{i,j} > 0$ or the product $\text{Support}_{i,j} \cdot \text{PageRank}_j$ for all pages. While the latter variant appears more sophisticated, in our experiments we find Variant A perform slightly better.

Algorithm 4: Irregularity Calculation for Web page i, two variants.

Create a list of buckets for $k = 0, 1, \ldots$
for all Web pages j with $\text{Support}_{i,j} > 0$ **do**
$r = \begin{cases} \text{PageRank}_j & \{\text{Variant A}\} \\ \text{Support}_{i,j} \cdot \text{PageRank}_j & \{\text{Variant B}\} \end{cases}$
Add j to bucket k with $a \cdot b^{k-1} < r \leq a \cdot b^k$ {we use $a = 0.1$, $b = 0.7$}
for all nonempty buckets $k = 0, 1, \ldots$ **do**
$X_k \leftarrow k, \quad Y_k \leftarrow \log(|\text{bucket}_i|)$
$\rho \leftarrow \text{Pearson-correlation}(X, Y)$

Given the PageRank histogram as above, we use a very simple approach to test its fit to a power law distribution. We split pages into buckets by PageRank; we let bucket boundary values grow exponentially as $a \cdot b^k$. We use $a = 0.1$ and $b = 0.7$; the choice has little effect on the results. If the PageRank values follow a power law distribution such that the ℓ-th page in order has rank proportional to $c \cdot \ell^{-\alpha}$, then the theoretic size of bucket k should be

$$\begin{aligned} \int_{a \cdot b^k}^{a \cdot b^{k+1}} c \cdot \ell^{-\alpha} \, d\ell &= c'(b^{k \cdot (-\alpha-1)} - b^{(k+1) \cdot (-\alpha-1)}) \\ &= c'' b^{k \cdot (-\alpha-1)}. \end{aligned}$$

Hence logarithm of the theoretical count within bucket k is linear in k. In our algorithm we penalize proportional to the Pearson correlation coefficient between the index and the logarithm of the count within the bucket as one possible measure for testing a line fit over the data.

4 Experiments

4.1 Data set

Torsten Suel and Yen-Yu Chen kindly provided us with the web graph and the set of URLs extracted from a 31.2 M page crawl of the `.de` domain. The crawl was carried out by the Polybot crawler [37] in April 2004. This German graph is denser than the usual web graphs, it has 962 M edges, which implies an average out-degree of 30.82.

While all the algorithms mentioned in Section 3 can be implemented both in external memory and in a distributed network of workstations, we applied a trivial graph compression method to speed our experiments up by using internal memory. The compressed graph size is 1.3 GB. Computing the Monte Carlo Personalized PageRank for all the nodes took 17 hours, creating the inverted PPR database required 4 hours and resulted in a 14 GB database. Determining SpamRank's personalization vector took 20 minutes, which was followed by another 20 minutes of PageRank computation by simple power method. All programs were ran on a single Pentium 4 3.0 GHz machine with Linux OS.

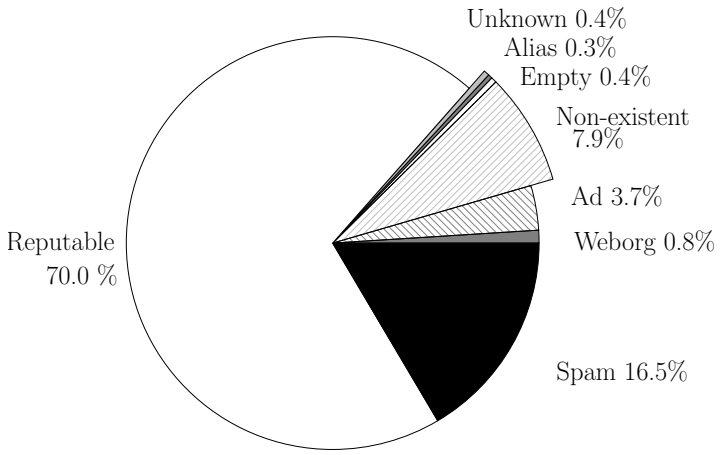


Figure 1: Distribution of categories in the evaluation sample

4.2 Results

To evaluate the signals given by SpamRank we chose a subset of the methods presented in [23]. Firstly we ordered the pages according to their PageRank value and assigned them to 20 consecutive buckets such that each bucket contained 5% of the total PageRank sum with bucket 1 containing the page with the highest PageRank. From each bucket we chose 50 URLs uniformly at random, resulting in a 1000 page sample heavily biased toward pages with high PageRank. Three of the authors received a 400 page subset of the sample, which we manually classified into one of the following categories: reputable, web organization, advertisement, spam, non-existent, empty, alias, and unknown (see [23] for detailed definition of the categories).

We observed a poor pairwise κ value [10] of 0.45 over the 100 pairs of common URLs. The majority of disagreements could be attributed to different rating of pages in affiliate programs and certain cliques. This shows that assessing link spam is nontrivial task for humans as well. By considering all remarks available concerning the judgements over the common set of URLs and using the experience gathered so far, one of the authors revised the classifications for the full 1000 page sample.

We observe relative fast changes among sample pages over time, in particular spam pages changing to non-existent. Judgements after final revision reflect the state as of April 2004.

Figure 1 shows the distribution of categories among the sample. Throwing away the non-existent, empty, unknown and alias pages gave us a usable 910 page sample. Note that the proportion of spam pages in our sample is somewhat higher than in previous studies [23, 18]. We attribute this to the denser web graph and the known notoriousness of spammers over the .de domain [19]. Because of the abundance of pages from the eBay.de domain – 0.8 M pages in the dataset – and due to the misuse of its affiliation program, we decided to treat pages from eBay.de as a separate subcategory in the next two figures.

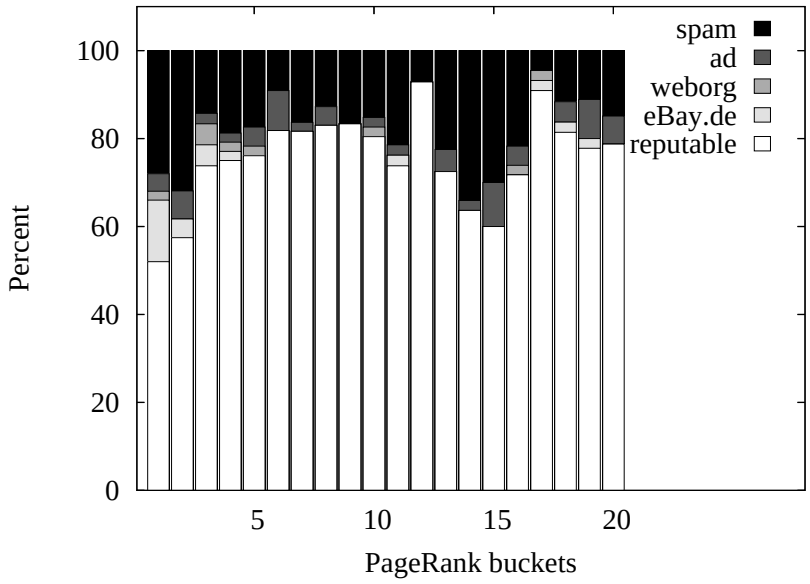


Figure 2: Distribution of categories among PageRank buckets

Figure 2 depicts the distribution of each category conditioned on the PageRank bucket. It can clearly be seen that a large amount of spam pages made it to the top two PageRank buckets where the percentage of spam is even higher than in the full sample.

Using the SpamRank ordering we again assigned each page to one of the 20 SpamRank buckets, the ith SpamRank bucket having exactly the same number of pages in it as the ith PageRank bucket. Figure 3 demonstrates that the first four SpamRank buckets contain a very large amount of spam; these buckets are low in truly reputable non-eBay content.

It is important to note that the ordering induced by SpamRank is very different from PageRank, therefore we cannot assess the properties of SpamRank using traditional precision/recall curves calculated over the original sample as it was drawn according to the PageRank distribution. We refrained from classifying a complete new sample according to the SpamRank distribution, instead we only drew a new random sample of 5×20 pages from the top 5 SpamRank buckets. As it can be seen in Figure 4 the top SpamRank buckets are rich in spam pages, though more than half of the pages in these buckets are not spam. Manual inspection of the non-spam pages revealed that they are predominantly pages from sites with dense, templatic internal structure such as forums, online retail catalogues and structured document archives. In terms of PageRank, the machine generated internal link structure of these sites behaves exactly like a large link farm, therefore we believe it is justified to mark them as pages with artificially inflated PageRank value.

Finally in Figure 5 we plotted the average difference between the PageRank and SpamRank bucket number of pages separately for spam and reputable pages (including eBay.de) in each PageRank bucket. The average demotion in SpamRank compared to PageRank is significantly higher for reputable pages. The (small) positive demotion for spam pages is explained by the fact that the top SpamRank buckets contain a number of fresh, either spammy or cliquish pages (see Figure 4) not included in the original sample.

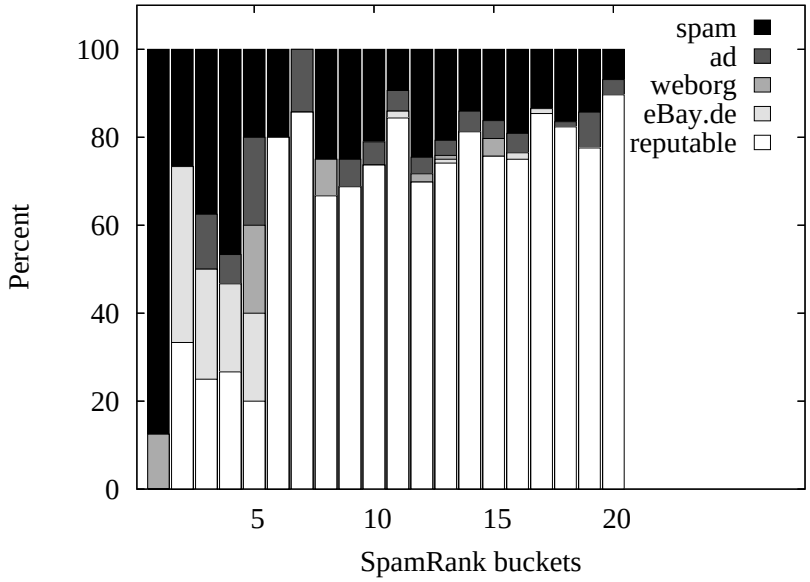


Figure 3: Distribution of categories among SpamRank buckets. Sampling is stratified using PageRank.

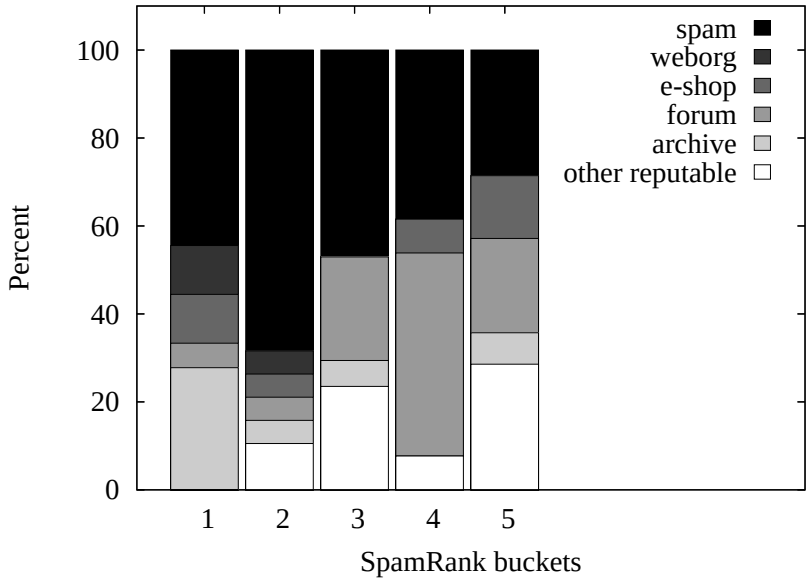


Figure 4: Distribution of categories among the top 5 SpamRank buckets. Sampling is stratified using SpamRank.

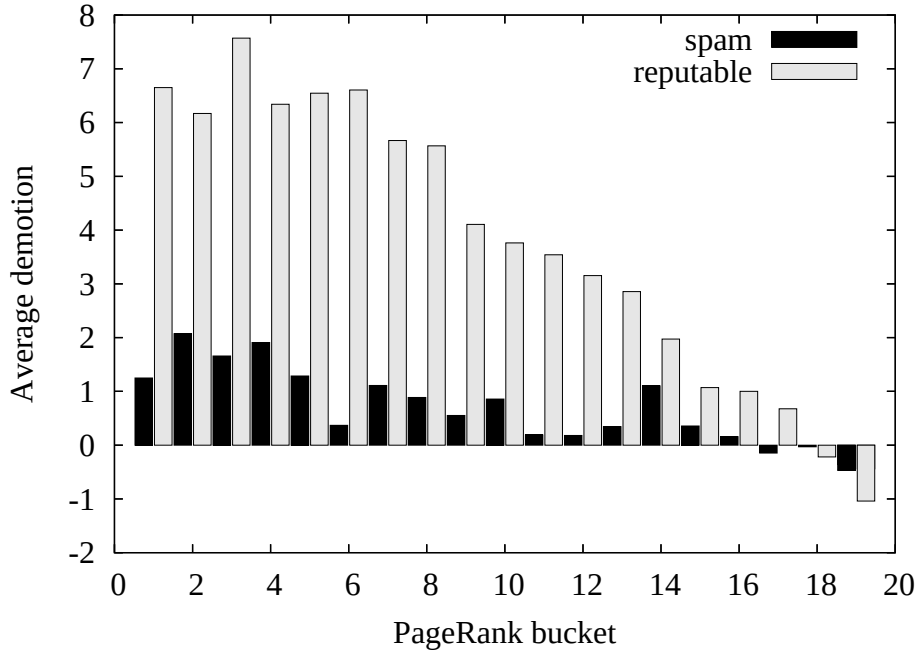


Figure 5: Demotion

5 Conclusions

We presented SpamRank, a three-stage, scalable Monte Carlo algorithm for computing a personalized PageRank vector biased toward link spam pages. Our experiments demonstrated that SpamRank is indeed capable of differentiating among spam and non-spam pages. A number of questions left to subsequent work are as follows. Explore the effects of parameters and assess variants of the algorithm (e.g. penalty dependent on the PageRank of a suspicious target page). Produce a ranking that retains the reputable part of PageRank. Incorporate SpamRank into a ranking function and measure its effect on precision for popular or financially lucrative queries. Lastly compare and evaluate SpamRank against Adaptive Epsilon [39] and Wu and Davison's method [38], the other publicly known PageRank schemes designed to be spam-resistant without human effort.

6 Acknowledgement

The authors would like to thank Torsten Suel and Yen-Yu Chen for providing the `.de` web graph, Alessandro Panconesi and Prabhakar Raghavan for inspiring our research at Bertinoro Web Bar 2004, and lastly Dániel Fogaras and Balázs Rácz for many fruitful discussions.

References

[1] BadRank as the opposite of PageRank. `http://en.pr10.info/pagerank0-badrank/`.

[2] The Word Spy - spamdexing. `http://www.wordspy.com/words/spamdexing.asp`.

[3] E. Amitay, D. Carmel, A. Darlow, R. Lempel, and A. Soffer. The Connectivity Sonar: Detecting site functionality by structural patterns. In *Proceedings of the 14th ACM Conference on Hypertext and Hypermedia (HT)*, Nottingham, United Kingdom, August 26-30 2003.

[4] R. Baeza-Yates, C. Castillo, and V. López. PageRank increase under different collusion topologies. In *Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2005.

[5] A.-L. Barabási, R. Albert, and H. Jeong. Scale-free characteristics of random networks: the topology of the word-wide web. *Physica A*, 281:69–77, 2000.

[6] K. Bharat and M. R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, pages 104–111, Melbourne, AU, 1998.

[7] M. Bianchini, M. Gori, and F. Scarselli. Inside PageRank. *ACM Transactions on Internet Technology*, 5(1), 2005.

[8] A. Borodin, G. O. Roberts, J. S. Rosenthal, and P. Tsaparas. Finding authorities and hubs from link structures on the world wide web. In *Proceedings of the 10th World Wide Web Conference (WWW)*, pages 415–429, 2001.

[9] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[10] J. Carletta. Assessing agreement on classification tasks: the kappa statistic. *Computational Linguistics*, 22(2):249–254, 1996.

[11] S. Chakrabarti, B. E. Dom, S. R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, D. Gibson, and J. M. Kleinberg. Mining the Web's link structure. *Computer*, 32(8):60–67, 1999.

[12] C. Chekuri, M. H. Goldwasser, P. Raghavan, and E. Upfal. Web search using automatic classification. In *Proceedings of the 6th International World Wide Web Conference (WWW)*, San Jose, US, 1997.

[13] B. D. Davison. Recognizing nepotistic links on the web. In *AAAI-2000 Workshop on Artificial Intelligence for Web Search, Austin, TX*, pages 23–28, July 30 2000.

[14] C. Ding, X. He, P. Husbands, H. Zha, and H. Simon. PageRank, HITS and a unified framework for link analysis. In *Proceedings of the 25th ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 353–354, Tampere, Finland, 2002.

[15] C. Dwork, S. R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th International World Wide Web Conference (WWW)*, pages 613–622, Hong Kong, 2001.

[16] D. H. eds. *Proceedings of the 1st Conference on Email and Anti-Spam (CEAS)*. Mountain View, CA, July 2004.

[17] N. Eiron, K. S. McCurley, and J. A. Tomlin. Ranking the web frontier. In *Proceedings of the 13th International World Wide Web Conference (WWW)*, pages 309–318, New York, NY, USA, 2004. ACM Press.

[18] D. Fetterly, M. Manasse, and M. Najork. Spam, damn spam, and statistics – Using statistical analysis to locate spam web pages. In *Proceedings of the 7th International Workshop on the Web and Databases (WebDB)*, Paris, France, 2004.

[19] D. Fetterly, M. Manasse, M. Najork, and J. Wiener. A large-scale study of the evolution of web pages. In *Proceedings of the 12th International World Wide Web Conference (WWW)*, Budapest, Hungary, 2003. ACM Press.

[20] D. Fogaras. Where to start browsing the web? In *Proceedings of the 3rd International Workshop on Innovative Internet Community Systems I2CS, published as LNCS 2877*, pages 65–79, 2003.

[21] D. Fogaras and B. Rácz. Towards scaling fully personalized PageRank. In *Proceedings of the 3rd Workshop on Algorithms and Models for the Web-Graph (WAW)*, pages 105–117, Rome, Italy, October 2004. Full version to appear in Internet Mathematics.

[22] Z. Gyöngyi and H. Garcia-Molina. Web spam taxonomy. In *Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2005.

[23] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with TrustRank. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB)*, Toronto, Canada, 2004.

[24] M. Henziger, R. Motwani, and C. Silverstein. Challenges in web search engines. *SIGIR Forum*, 36(2), 2002.

[25] G. Jeh and J. Widom. Scaling personalized web search. In *Proceedings of the 12th International World Wide Web Conference (WWW)*, pages 271–279, Budapest, Hungary, 2003. ACM Press.

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

[27] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Stochastic models for the web graph. In *Proceedings of the 41st IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, 2000.

[28] S. K. Lam and J. Riedl. Shilling recommender systems for fun and profit. In *Proceedings of the 13th International World Wide Web Conference (WWW)*, pages 393–402, New York, NY, USA, 2004. ACM Press.

[29] A. N. Langville and C. D. Meyer. Deeper inside PageRank. *Internet Mathematics*, 1(3):335–400, 2004.

[30] L. Laura, S. Leonardi, S. Millozzi, U. Meyer, and Y. F. Sibeyn. Algorithms and experiments for the Web graph. In *Proceedings of the European Symposium on Algorithms*, 2003.

[31] R. Lempel and S. Moran. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. In *Proceedings of the 9th World Wide Web Conference (WWW)*, 2000.

[32] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford University, 1998.

[33] G. Pandurangan, P. Raghavan, and E. Upfal. Using PageRank to Characterize Web Structure. In *8th Annual International Computing and Combinatorics Conference (COCOON)*, 2002.

[34] A. Perkins. White paper: The classification of search engine spam, Sept. 2001. Online at http://www.silverdisc.co.uk/articles/spam-classification/.

[35] G. O. Roberts and J. S. Rosenthal. Downweighting tightly knit communities in world wide web rankings. *Advances and Applications in Statistics (ADAS)*, 3:199–216, 2003.

[36] A. Singhal. Challenges in running a commercial search engine. In *IBM Search and Collaboration Seminar 2004*. IBM Haifa Labs, 2004.

[37] T. Suel and V. Shkapenyuk. Design and implementation of a high-performance distributed web crawler. In *Proceedings of the IEEE International Conference on Data Engineering*, February 2002.

[38] B. Wu and B. D. Davison. Identifying link farm pages. In *Proceedings of the 14th International World Wide Web Conference (WWW)*, 2005.

[39] H. Zhang, A. Goel, R. Govindan, K. Mason, and B. V. Roy. Making eigenvector-based reputation systems robust to collusion. In *Proceedings of the 3rd Workshop on Algorithms and Models for the Web-Graph (WAW)*, Rome, Italy, October 2004. Full version to appear in Internet Mathematics.

Web Spam Taxonomy

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

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

Abstract

Web spamming refers to actions intended to mislead search engines into ranking some pages higher than they deserve. Recently, the amount of web spam has increased dramatically, leading to a degradation of search results. This paper presents a comprehensive taxonomy of current spamming techniques, which we believe can help in developing appropriate countermeasures.

1 Introduction

As more and more people rely on the wealth of information available online, increased exposure on the World Wide Web may yield significant financial gains for individuals or organizations. Most frequently, search engines are the entryways to the Web; that is why some people try to mislead search engines, so that their pages would rank high in search results, and thus, capture user attention.

Just as with emails, we can talk about the phenomenon of *spamming* the Web. The primary consequence of web spamming is that the quality of search results decreases. For instance, at the time of writing this article, the second result returned by a major search engine for the query "Kaiser pharmacy" was a page on the spam web site `techdictionary.com`. This site contains only a few lines of useful information (mainly some term definitions, probably copied from a real dictionary), but consists of thousands of pages, each repeating the same content and pointing to dozens of other pages. All the pages were probably created to boost the rankings of some others, and none of them seems to be particularly useful for anyone looking for pharmacies affiliated with Kaiser-Permanente.

The secondary consequence of spamming is that search engine indexes are inflated with useless pages, increasing the cost of each processed query.

To provide low-cost, quality services, it is critical for search engines to address web spam. Search engines currently fight spam with a variety of often manual techniques, but as far as we know, they still lack a fully effective set of tools for combating it. We believe that the first step in combating spam is *understanding* it, that is, analyzing the techniques the spammers use to mislead search engines. A proper understanding of spamming can then guide the development of appropriate countermeasures.

To that end, in this paper we organize web spamming techniques into a taxonomy that can provide a framework for combating spam. We also provide an overview of published statistics about web spam to underline the magnitude of the problem.

There have been brief discussions of spam in the scientific literature [3, 6, 12]. One can also find details for several specific techniques on the Web itself (e.g., [11]). Nevertheless, we believe that this paper offers the first comprehensive taxonomy of all important spamming techniques known to date. To build our taxonomy, we worked closely with experts at one of the major search engine companies, relying on their experience, while at the same time investigating numerous spam instances on our own.

Some readers might question the wisdom of revealing spamming secrets, concerned that this might encourage additional spamming. We assure readers that nothing in this paper is secret to the spammers; it is only most of the web users who are unfamiliar with the techniques presented here. We believe that by publicizing these spamming techniques we will raise the awareness and interest of the research community.

2 Definition

The objective of a search engine is to provide high-quality results by correctly identifying all web pages that are *relevant* for a specific query, and presenting the user with some of the most *important* of those relevant pages. Relevance is usually measured through the textual similarity between the query and a page. Pages can be given a query-specific, numeric relevance score; the higher the number, the more relevant the

page is to the query. Importance refers to the global (query-independent) popularity of a page, as often inferred from the link structure (e.g., pages with many incoming links are more important), or perhaps other indicators. In practice, search engines usually combine relevance and importance, computing a combined *ranking* score that is used to order query results presented to the user.

We use the term *spamming* (also, *spamdexing*) to refer to any deliberate human action that is meant to trigger an unjustifiably favorable relevance or importance for some web page, considering the page's true value. We will use the adjective *spam* to mark all those web objects (page content items or links) that are the result of some form of spamming. People who perform spamming are called *spammers.*

One can locate on the World Wide Web a handful of other definitions of web spamming. For instance, some of the definitions (e.g., [13]) are close to ours, stating that any modification done to a page solely because search engines exist is spamming. Specific organizations or web user groups (e.g., [9]) define spamming by enumerating some of the techniques that we present in Sections 3 and 4.

An important voice in the web spam arena is that of *search engine optimizers* (SEOs), such as SEO Inc. (`www.seoinc.com`) or Bruce Clay (`www.bruceclay.com`). The activity of some SEOs benefits the whole web community, as they help authors create well-structured, high-quality pages. However, most SEOs engage in practices that we call spamming. For instance, there are SEOs who define spamming exclusively as increasing relevance for queries not related to the topic(s) of the page. These SEOs endorse and practice techniques that have an impact on importance scores, to achieve what they call "ethical" web page positioning or optimization. Please note that according to our definition, all types of actions intended to boost ranking (either relevance, or importance, or both), without improving the true value of a page, are considered spamming.

There are two categories of techniques associated with web spam. The first category includes the boosting techniques, i.e., methods through which one seeks to achieve high relevance and/or importance for some pages. The second category includes hiding techniques, methods that by themselves do not influence the search engine's ranking algorithms, but that are used to hide the adopted boosting techniques from the eyes of human web users. The following two sections discuss each of these two categories in more detail.

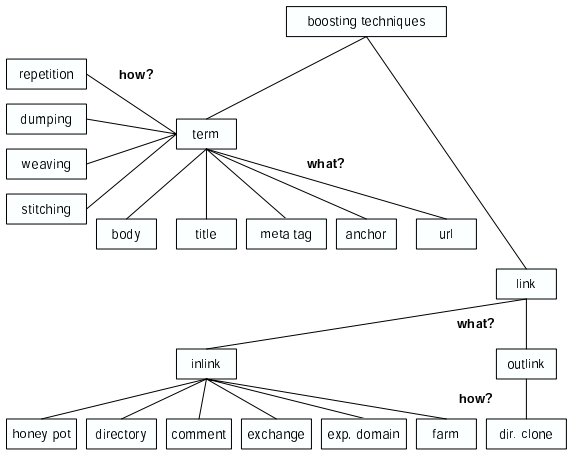


Figure 1: Boosting techniques.

3 Boosting Techniques

In this section we present spamming techniques that influence the ranking algorithms used by search engines. Figure 1 depicts our taxonomy, in order to guide our discussion.

3.1 Term Spamming

In evaluating textual relevance, search engines consider where on a web page query terms occurs. Each type of location is called a *field*. The common text fields for a page p are the document body, the title, the meta tags in the HTML header, and page p's URL. In addition, the anchor texts associated with URLs that point to p are also considered belonging to page p (anchor text field), since they often describe very well the content of p. The terms in p's text fields are used to determine the relevance of p with respect to a specific query (a group of query terms), often with different weights given to different fields. *Term spamming* refers to techniques that tailor the contents of these text fields in order to make spam pages relevant for some queries.

3.1.1 Target Algorithms

The algorithms used by search engines to rank web pages based on their text fields use various forms of the fundamental TFIDF metric used in information retrieval [1]. Given a specific text field, for each term t that is common for the text field and a query, TF(t) is the frequency of that term in the text field. For instance, if the term "apple" appears 6 times in the document body that is made up of a total of 30 terms, TF("apple") is $6/30 = 0.2$. The inverse document frequency IDF(t) of a term t is related to the number

of documents in the collection that contain t. For instance, if "apple" appears in 4 out of the 40 documents in the collection, its IDF("apple") score will be 10. The TFIDF score of a page p with respect to a query q is then computed over all common terms t:

$$\text{TFIDF}(p, q) = \sum_{t \in p \text{ and } t \in q} \text{TF}(t) \cdot \text{IDF}(t)$$

With TFIDF scores in mind, spammers can have two goals: either to make a page relevant for a large number of queries (i.e., to receive a non-zero TFIDF score), or to make a page very relevant for a specific query (i.e., to receive a high TFIDF score). The first goal can be achieved by including a large number of distinct terms in a document. The second goal can be achieved by repeating some "targeted" terms. (We can assume that spammers cannot have real control over the IDF scores of terms. Moreover, some search engines ignore IDF scores altogether. Thus, the primary way of increasing the TFIDF scores is by increasing the frequency of terms within specific text fields of a page.)

3.1.2 Techniques

Term spamming techniques can be grouped based on the text field in which the spamming occurs. Therefore, we distinguish:

- *Body spam.* In this case, the spam terms are included in the document body. This spamming technique is among the simplest and most popular ones, and it is almost as old as search engines themselves.

- *Title spam.* Today's search engines usually give a higher weight to terms that appear in the title of a document. Hence, it makes sense to include the spam terms in the document title.

- *Meta tag spam.* The HTML meta tags that appear in the document header have always been the target of spamming. Because of the heavy spamming, search engines currently give low priority to these tags, or even ignore them completely. Here is a simple example of a spammed keywords meta tag:

  <meta name="keywords" content="buy, cheap, cameras, lens, accessories, nikon, canon">

- *Anchor text spam.* Just as with the document title, search engines assign higher weight to anchor text terms, as they are supposed to offer a summary of the pointed document. Therefore, spam terms are sometimes included in the anchor text of the HTML hyperlinks to a page. Please note that this spamming technique is different from the previous ones, in the sense that the spam terms are added not to a target page itself, but the other pages that point to the target. As anchor text gets indexed for both pages, spamming it has impact on the ranking of both the source and target pages. A simple anchor text spam is:

  free, great deals, cheap, inexpensive, cheap, free

- *URL spam.* Some search engines also break down the URL of a page into a set of terms that are used to determine the relevance of the page. To exploit this, spammers sometimes create long URLs that include sequences of spam terms. For instance, one could encounter spam URLs like:

  buy-canon-rebel-20d-lens-case.camerasx.com,
  buy-nikon-d100-d70-lens-case.camerasx.com,
  ...

  Some spammers even go to the extent of setting up a DNS server that resolves *any* host name within a domain.

Often, spamming techniques are combined. For instance, anchor text and URL spam is often encountered together with link spam, which will be discussed in Section 3.2.2.

Another way of grouping term spamming techniques is based on the type of terms that are added to the text fields. Correspondingly, we have:

- *Repetition* of one or a few specific terms. This way, spammers achieve an increased relevance for a document with respect to a small number of query terms.

- *Dumping* of a large number of unrelated terms, often even entire dictionaries. This way, spammers make a certain page relevant to many different queries. Dumping is effective against queries that include relatively rare, obscure terms: for such queries, it is probable that only a couple of pages are relevant, so even a spam page with a low relevance/importance would appear among the top results.

- *Weaving* of spam terms into copied contents. Sometimes spammers duplicate text corpora (e.g., news articles) available on the Web and insert spam terms into them at random positions. This technique is effective if the topic of the original real text was so rare that only a small number of relevant pages exist. Weaving is also used for *dilution*, i.e., to conceal some repeated spam terms

within the text, so that search engine algorithms that filters out plain repetition would be misled. A short example of spam weaving is:

> Remember not only airfare to say the right plane tickets thing in the right place, but far cheap travel more difficult still, to leave hotel rooms unsaid the wrong thing at vacation the tempting moment.

- *Phrase stitching* is also used by spammers to create content quickly. The idea is to glue together sentences or phrases, possibly from different sources; the spam page might then show up for queries on any of the topics of the original sentences. For instance, a spammer using this paper as source could come up with the following collage:

> The objective of a search engine is to provide high-quality results by correctly identifying. Unjustifiably favorable boosting techniques, i.e., methods through which one seeks relies on the identification of some common features of spam pages.

3.2 Link Spamming

Beside term-based relevance metrics, search engines also rely on link information to determine the importance of web pages. Therefore, spammers often create link structures that they hope would increase the importance of one or more of their pages.

3.2.1 Target Algorithms

For our discussion of the algorithms targeted by link spam, we will adopt the following model. For a spammer, there are three types of pages on the Web:

1. *Inaccessible* pages are those that a spammer cannot modify. These are the pages out of reach; the spammer cannot influence their outgoing links. (Note that a spammer can still point to inaccessible pages.)

2. *Accessible* pages are maintained by others (presumably not affiliated with the spammer), but can still be modified in a limited way by a spammer. For example, a spammer may be able to post a comment to a blog entry, and that comment may contain a link to a spam site. As infiltrating accessible pages is usually not straightforward, let us say that a spammer has a limited budget of m accessible pages. For simplicity, we assume that at most one outgoing link can be added to each accessible page.

3. *Own* pages are maintained by the spammer, who thus has full control over their contents. We call the group of own pages a *spam farm* Σ. A spammer's goal is to boost the importance of one or more of his or her own pages. For simplicity, say there is a single target page t. There is a certain maintenance cost (domain registration, web hosting) associated with a spammer's own pages, so we can assume that a spammer has a limited budget of n such pages, not including the target page.

With this model in mind, we discuss the two well-known algorithms used to compute importance scores based on link information.

HITS. The original HITS algorithm was introduced in [7] to rank pages on a specific topic. It is more common, however, to use the algorithm on all pages on the Web to assigns global *hub* and *authority* scores to each page. According to the circular definition of HITS, important hub pages are those that *point to* many important authority pages, while important authority pages are those *pointed to* by many hubs. A search engine that uses the HITS algorithm to rank pages returns as query result a blending of the pages with the highest hub and authority scores.

Hub scores can be easily spammed by adding outgoing links to a large number of well known, reputable pages, such as `www.cnn.com` or `www.mit.edu`. Thus, a spammer should add many outgoing links to the target page t to increase its hub score.

Obtaining a high authority score is more complicated, as it implies having many incoming links from presumably important hubs. A spammer could boost the hub scores of his n pages (once again, by adding many outgoing links to them) and then make those pages point to the target. Links from important accessible hubs could increase the target's authority score even further. Therefore, the rule here is "the more the better": within the limitations of the budget, the spammer should have all own and accessible pages point to the target. Non-target own pages should also point to as many other (known important) authorities as possible.

PageRank. PageRank, as described in [10], uses incoming link information to assign global importance scores to all pages on the Web. It assumes that the number of incoming links to a page is related to that page's popularity among average web users (people would point to pages that they find important). The intuition behind the algorithm is that a web page is important if several other important web pages point to it. Correspondingly, PageRank is based on a mutual reinforcement between pages: the importance of a certain page *influences* and is *being influenced* by the importance of some other pages.

A recent analysis of the algorithm [2] showed that the total PageRank score $\mathrm{PR}(\Gamma)$ of a group Γ of pages

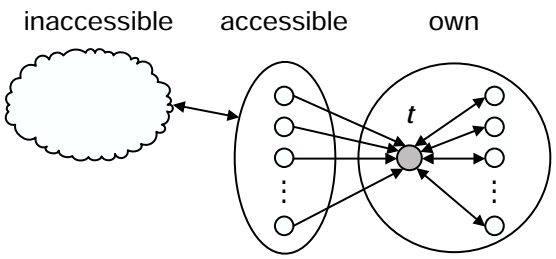


Figure 2: An optimal link structure for PageRank.

(at the extreme, a single page) depends on four factors:

$$\mathrm{PR}(\Gamma) = \mathrm{PR}_{\mathrm{static}}(\Gamma) + \mathrm{PR}_{\mathrm{in}}(\Gamma) - \mathrm{PR}_{\mathrm{out}}(\Gamma) - \mathrm{PR}_{\mathrm{sink}}(\Gamma),$$

where $\mathrm{PR}_{\mathrm{static}}$ is the score component due to the static score distribution (random jump); $\mathrm{PR}_{\mathrm{in}}$ is the score received through the incoming links from external pages; $\mathrm{PR}_{\mathrm{out}}$ is the score leaving Γ through the outgoing links to external pages; and $\mathrm{PR}_{\mathrm{sink}}$ is the score loss due to those pages within the group that have no outgoing links.

For our spam farm model, the previous formula leads to a class of optimal link structures that were proved to maximize the score of the target page [4]. One such optimal structure is presented in Figure 2; it has the arguably desirable properties that (1) it makes all own pages reachable from the accessible ones (so that they could be crawled by a search engine), and (2) it does this using a minimal number of links. We can observe how the presented structure maximizes the total PageRank score of the spam farm, and of page t in particular:

1. All available n own pages are part of the spam farm, maximizing the static score $\mathrm{PR}_{\mathrm{static}}(\Sigma)$.
2. All m accessible pages point to the spam farm, maximizing the incoming score $\mathrm{PR}_{\mathrm{in}}(\Sigma)$.
3. Links pointing outside the spam farm are suppressed, making $\mathrm{PR}_{\mathrm{out}}(\Sigma)$ equal to zero.
4. All pages within the farm have some outgoing links, rendering a zero $\mathrm{PR}_{\mathrm{sink}}(\Sigma)$ score component.

Within the spam farm, the the score of page t is maximal because:

1. All accessible and own pages point directly to the target, maximizing its incoming score $\mathrm{PR}_{\mathrm{in}}(t)$.
2. The target points to all other own pages. Without such links, t would had lost a significant part of its score ($\mathrm{PR}_{\mathrm{sink}}(t) > 0$), and the own pages would had been unreachable from outside the spam farm. Note that it would not be wise to add links from the target to pages outside the farm, as those would decrease the total PageRank of the spam farm.

As we can see in Figure 2, the "more is better" rule also applies to PageRank. It is true that setting up sophisticated link structures within a spam farm does not improve the ranking of the target page. However, a spammer can achieve high PageRank by accumulating many incoming links from accessible pages, and/or by creating large spam farms with all the pages pointing to the target. The corresponding spamming techniques are presented next.

3.2.2 Techniques

We group link spamming techniques based on whether they add numerous outgoing links to popular pages or they gather many incoming links to a single target page or group of pages.

Outgoing links. A spammer might manually add a number of outgoing links to well-known pages, hoping to increase the page's hub score. At the same time, the most wide-spread method for creating a massive number of outgoing links is *directory cloning:* One can find on the World Wide Web a number of directory sites, some larger and better known (e.g., the DMOZ Open Directory, `dmoz.org`, or the Yahoo! directory, `dir.yahoo.com`), some others smaller and less famous (e.g., the Librarian's Index to the Internet, `lii.org`). These directories organize web content around topics and subtopics, and list relevant sites for each. Spammers then often simply replicate some or all of the pages of a directory, and thus create massive outgoing-link structures quickly.

Incoming links. In order to accumulate a number of incoming links to a single target page or set of pages, a spammer might adopt some of the following strategies:

- *Create a honey pot*, a set of pages that provide some useful resource (e.g., copies of some Unix documentation pages), but that also have (hidden) links to the target spam page(s). The honey pot then attracts people to point to it, boosting indirectly the ranking of the target page(s). Please note that the previously mentioned directory clones could act as honey pots.
- *Infiltrate a web directory.* Several web directories allow webmasters to post links to their sites under some topic in the directory. It might happen that

the editors of such directories do not control and verify link additions strictly, or get misled by a skilled spammer. In these instances, spammers may be able to add to directory pages links that point to their target pages. As directories tend to have both high PageRank and hub scores, this spamming technique is useful in boosting both the PageRank and authority scores of target pages.

- *Post links on blogs, unmoderated message boards, guest books, or wikis.* As mentioned earlier in Section 3.2.1, spammers may include URLs to their spam pages as part of the seemingly innocent comments/messages they post. Without an editor or a moderator to oversee all submitted comments/messages, pages of the blog, message board, or guest book end up linking to spam. Even if there is an editor or a moderator, it could be nontrivial to detect spam comments/messages as they might employ some of the hiding techniques presented in the next section. Here is a simple example of a spam blog comment that features both link and anchor text spamming:

  Nice story. Read about my Las Vegas casino trip.

  It is important to mention that blog comment spamming is gaining popularity, and it is not only a problem for search engines, but also has a strong direct influences on the large community of millions of bloggers: for the web users with their own blogs, comment spamming represents a nuisance similar to email spamming. Recently, a number of tools and initiatives were launched to curb comment spamming. For instance, some bloggers maintain lists of domain names that appear in spam URLs [8].

- *Participate in link exchange.* Often times, a group of spammers set up a link exchange structure, so that their sites point to each other.

- *Buy expired domains.* When a domain names expires, the URLs on various other web sites that point to pages within the expired domain linger on for some time. Some spammers buy expired domains and populate them with spam that takes advantage of the false relevance/importance conveyed by the pool of old links.

- *Create own spam farm.* These days spammers can control a large number of sites and create arbitrary link structures that would boost the ranking of some target pages. While this approach was prohibitively expensive a few years ago, today it is very common as the costs of domain registration and web hosting have declined dramatically.

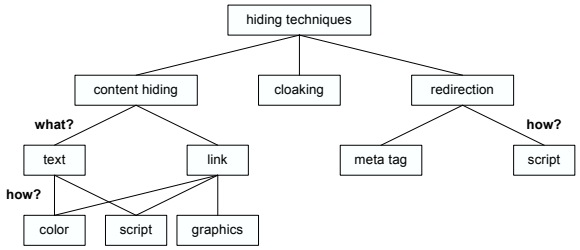


Figure 3: Spam hiding techniques.

4 Hiding Techniques

It is usual for spammers to conceal the telltale signs (e.g., repeated terms, long lists of links) of their activities. They use a number of techniques to hide their abuse from regular web users visiting spam pages, or from the editors at search engine companies who try to identify spam instances. This section offers an overview of the most common spam hiding techniques, also summarized in Figure 3.

4.1 Content Hiding

Spam terms or links on a page can be made invisible when the browser renders the page. One common technique is using appropriate *color schemes*: terms in the body of an HTML document are not visible if they are displayed in the same color as the background. Color schemes can be defined either in the HTML document or in an attached cascading style sheet (CSS). We show a simple HTML example next:

```
<body background="white">
  <font color="white">hidden text</font>
  ...
</body>
```

In a similar fashion, spam links can be hidden by avoiding anchor text. Instead, spammers often create tiny, 1×1-pixel *anchor images* that are either transparent or background-colored:

```
<a href="target.html"><img src="tinyimg.gif"></a>
```

A spammer can also use *scripts* to hide some of the visual elements on the page, for instance, by setting the visible HTML style attribute to false.

4.2 Cloaking

If spammers can clearly identify web crawler clients, they can adopt the following strategy, called *cloaking*: given a URL, spam web servers return one specific HTML document to a regular web browser, while they return a different document to a web crawler. This way, spammers can present the ultimately intended content to the web users (without traces of spam on the page), and, at the same time, send a spammed document to the search engine for indexing.

The identification of web crawlers can be done in two ways. On one hand, some spammers maintain a list of IP addresses used by search engines, and identify web crawlers based on their matching IPs. On the other hand, a web server can identify the application requesting a document based on the user-agent field in the HTTP request message. For instance, in the following simple HTTP request message the user-agent name is that one used by the Microsoft Internet Explorer 6 browser:

```
GET /db_pages/members.html HTTP/1.0
Host: www-db.stanford.edu
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
```

The user-agent names are not strictly standardized, and it is really up to the requesting application what to include in the corresponding message field. Nevertheless, search engine crawlers identify themselves by a name distinct from the ones used by traditional web browser applications. This is done in order to allow webmasters to block access to some of the contents, control network traffic parameters, or even perform some well-intended, legitimate optimizations. For instance, a few sites serve to search engines versions of their pages that are free from navigational links, advertisements, and other visual elements related to the presentation, but not to the content. This kind of activity might even be welcome by some of the search engines, as it helps indexing the useful information.

4.3 Redirection

Another way of hiding the spam content on a page is by automatically *redirecting* the browser to another URL as soon as the page is loaded. This way the page still gets indexed by the search engine, but the user will not ever see it—pages with redirection act as *intermediates* (or *proxies*, *doorways*) for the ultimate targets, which spammers try to serve to a user reaching their sites through search engines.

Redirection can be achieved in a number of ways. A simple approach is to take advantage of the refresh meta tag in the header of an HTML document. By setting the refresh time to zero and the refresh URL to the target page, spammers can achieve redirection as soon as the page gets loaded into the browser:

```
<meta http-equiv="refresh" content=
"0;url=target.html">
```

While the previous approach is not hard to implement, search engines can easily identify such redirection attempts by parsing the meta tags. More sophisticated spammers achieve redirection as part of some script on the page, as scripts are not executed by the crawlers:

```
<script language="javascript"><!--
   location.replace("target.html")
--></script>
```

5 Statistics

While we have a good understanding of spamming techniques, the publicly available statistical data describing the amount and nature of web spam is very limited. In this section we review some of what is known.

Two papers discuss the prevalence of web spam, presenting results from three experiments. Fetterly *et al.* [3] manually evaluated sample pages from two different data sets.

The first data set (DS1) represented 150 million URLs that were crawled repeatedly, once every week over a period of 11 weeks, from November 2002 to February 2003. The authors retained 0.1% of all crawled pages, chosen based on a hash of the URLs. A manual inspection of 751 pages sampled from the set of retained pages yielded 61 spam pages, indicating a prevalence of 8.1% spam in the data set, with a confidence interval of 1.95% at 95% confidence.

The second data set (DS2) was the result of a single breadth-first search started at the Yahoo! home page, conducted between July and September 2002. The search covered about 429 million pages. During a later manual evaluation, from a random sample of 1,000 URLs, the authors were able to download 535 pages, of which 37 (6.9%) were spam.

A third, independent set of statistics is provided by Gyöngyi *et al.* [5]. In this case, the authors used the complete set of pages crawled and indexed by the AltaVista search engine as of August 2003. The several billion web pages were grouped into approximately 31 million web sites (DS3), each corresponding roughly to an individual web host. Instead of random sampling, the following strategy was adopted: the authors segmented the list of sites in decreasing PageRank order

into 20 buckets. Each of the buckets contained a different number of sites, with PageRank scores summing up to 5 percent of the total PageRank. Accordingly, the first bucket contained the 86 sites with the highest PageRank scores, bucket 2 the next 665, while the last bucket contained 5 million sites that were assigned the lowest PageRank scores. The upper part of Figure 4 shows the size of each bucket on a logarithmic scale.

First, an initial sample of 1000 sites was constructed by selecting 50 sites at random from each bucket. Then, the sample was reduced to 748 existing sites that could be categorized clearly. A manual inspection discovered that 135 (18%) of these sites were spam. The lower part of Figure 4 presents the fraction of spam in each bucket. It is interesting to note that almost 20% of the second PageRank bucket is spam, indicating that some sophisticated spammers can achieve high importance scores. Also, note that there is a high prevalence of spam (almost 50%) in buckets 9 and 10. This fact seems to indicate that "average" spammers can generate a significant amount of spam with mid-range logarithmic PageRank.

Table 1 summarizes the results from the three presented experiments. The differences between the reported prevalence figures could be due to an interplay of several factors:

- The crawls were performed at different times. It is possible that the amount of spam increased over time.
- Different crawling strategies were used.
- There could be a difference between the fraction of sites that are spam and the fraction of pages that are spam. In other words, it could be the case that the average number of pages per site is different for spam and non-spam sites.
- Classification of spam could be subjective; individuals may have broader or narrower definition of what constitutes spam.

Despite the discrepancies, we can probably safely estimate that 10-15% of the content on the Web is spam.

As the previous discussion illustrates, our statistical knowledge of web spam is sparse. It would be of interest to have data not only on what fraction of pages or sites is spam, but also on the relative sizes (as measured in bytes) of spam and non-spam on the Web. This would help us estimate what fraction of a search engine's resources (disk space, crawling/indexing/query processing time) is wasted on spam. Another important question is how spam evolves over time. Finally, we do not yet know much about the relative frequencies of different spamming techniques, and the co-occurrence patterns between them. It is suspected that currently almost all spammers use link spamming, usually combined with anchor text spamming, but there are no published research results supporting this hypothesis. It is our hope that future research in the field will provide some of the answers.

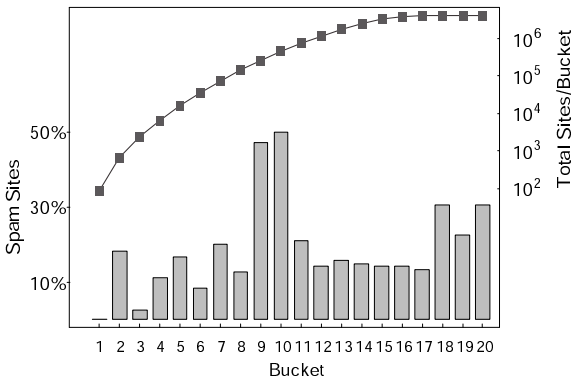


Figure 4: Bucket sizes and spam/bucket for DS3.

6 Conclusions

In this paper we presented a variety of commonly used web spamming techniques, and organized them into a taxonomy. We argue that such a structured discussion of the subject is important to raise the awareness of the research community. Our spam taxonomy naturally leads to a similar taxonomy of countermeasures. Correspondingly, we outline next the two approaches that a search engine can adopt in combating spam.

On one hand, it is possible to address each of the boosting and hiding technique presented in Sections 3 and 4 separately. Accordingly, one could:

1. *Identify* instances of spam, i.e., find pages that contain specific types of spam, and stop crawling and/or indexing such pages. Search engines usually take advantage of a group of automatic or semi-automatic, proprietary spam detection algorithms and the expertise of human editors to pinpoint and remove spam pages from their indexes. For instance, the techniques presented in [3] could be used to identify some of the spam farms with machine-generated structure/content.

2. *Prevent* spamming, that is, making specific spamming techniques impossible to use. For instance, a search engine's crawler could identify itself as a regular web browser application in order to avoid cloaking.

3. *Counterbalance* the effect of spamming. Today's

Data set	Crawl date	Data set size	Sample size	Spam
DS1	11/02 – 02/03	150 million pages	751 pages	8.1% of pages
DS2	07/02 – 09/02	429 million pages	535 pages	6.9% of pages
DS3	08/03	31 million sites	748 sites	18% of sites

Table 1: Spam prevalence statistics.

search engines use variations of the fundamental ranking methods (discussed in Sections 3.1.1 and 3.2.1) that feature some degree of spam resilience.

On the other hand, it is also possible to address the problem of spamming as a whole, despite the differences among individual spamming techniques. This approach relies on the identification of some common features of spam pages. For instance, the spam detection methods presented in [5] take advantage of the *approximate isolation* of reputable, non-spam pages: reputable web pages seldom point to spam. Thus, adequate link analysis algorithms can be used to separate reputable pages from any form of spam, without dealing with each spamming technique individually.

Acknowledgement

This paper is the result of many interesting discussions with one of our collaborators at a major search engine company, who wishes to remain anonymous. We would like to thank this person for the explanations and examples that helped us shape the presented taxonomy of web spam.

References

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

[2] Monica Bianchini, Marco Gori, and Franco Scarselli. Inside PageRank. *ACM Transactions on Internet Technology*, 5(1), 2005.

[3] Dennis Fetterly, Mark Manasse, and Marc Najork. Spam, damn spam, and statistics. In *Proceedings of the Seventh International Workshop on the Web and Databases (WebDB)*, 2004.

[4] Zoltán Gyöngyi and Hector Garcia-Molina. Link spam alliances. Technical report, Stanford University, 2005.

[5] Zoltán Gyöngyi, Hector Garcia-Molina, and Jan Pedersen. Combating web spam with TrustRank. In *Proceedings of the 30th International Conference on Very Large Databases (VLDB)*, 2004.

[6] Monika Henzinger, Rajeev Motwani, and Craig Silverstein. Challenges in web search engines. *SIGIR Forum*, 36(2), 2002.

[7] Jon Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5), 1999.

[8] Mark Carey Consulting Inc. Blog spam database. `http://www.markcarey.com/spamdb/`.

[9] Open Directory Project. Open directory editorial guidelines: Spamming. `http://dmoz.org/guidelines/spamming.html`.

[10] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.

[11] Alan Perkins. The classification of search engine spam. `http://www.ebrandmanagement.com/whitepapers/spam-classification/`.

[12] Mehran Sahami, Vibhu Mittal, Shumeet Baluja, and Henry Rowley. The happy searcher: Challenges in web information retrieval. In *Trends in Artificial Intelligence, 8th Pacific Rim International Conference on Artificial Intelligence (PRICAI)*, 2004.

[13] Shari Thurow. The search engine spam police, 2002. `http://www.searchenginewatch.com/searchday/article.php/2159061`.

An Analysis of Factors Used in Search Engine Ranking

Albert Bifet
Technical University of Catalonia
abifet@lsi.upc.es

Carlos Castillo
University of Chile
ccastillo@dcc.uchile.cl

Paul-Alexandru Chirita
L3S Research Center
chirita@l3s.de

Ingmar Weber
Max-Planck-Institute for Computer Science
iweber@mpi-sb.mpg.de

April 8, 2005

Abstract

This paper investigates the influence of different page features on the ranking of search engine results. We use Google (via its API) as our testbed and analyze the result rankings for several queries of different categories using statistical methods. We reformulate the problem of learning the underlying, hidden scores as a binary classification problem. To this problem we then apply both linear and non-linear methods. In all cases, we split the data into a training set and a test set to obtain a meaningful, unbiased estimator for the quality of our predictor. Although our results clearly show that the scoring function cannot be approximated well using only the observed features, we do obtain many interesting insights along the way and discuss ways of obtaining a better estimate and main limitations in trying to do so.

1 Introduction

In the age of digitalized information the world is relying more and more on web search engines when it comes to satisfying an information need. When given a new task, 88% of the time internet users start at such a search engine [11]. Part of the search engines' success might be due to their simplicity: you enter some words and the results are then output in form of a ranked list, in which the search engine estimates the relevance of each indexed website to your query.

Today Google claims to have indexed 8,058,044,651 web pages[1]. Even with this sheer enormous volume of information it is relatively "easy" to find a list of pages containing given query terms. The difficult part is then to select, from the possible myriad of matching pages, the "best" 10 or 20 according to some computable quality measure which, ideally, closely resembles the user's notion of relevance. The ability of any search engine to closely match this human notion has a major impact on its success.

Users are usually satisfied when the presented ranking leads to an answer to their queries. However, they normally do not question *why* they were presented with exactly this ranking, and how the search engine inferred which website is most relevant to their particular request. This situation is usually different when querying for their own name(s), as one might suddenly wonder why her personal homepage is (not) ranked highest.

E-commerce websites, on the other hand, have a much more tangible interest in the exact ranking process, as a high ranking in a popular search engine for a certain product query leads to more web traffic and ultimately translates into higher sales. Thus, "Search Engine Optimization" (SEO) has long been recognized as a lucrative business on its own.

Although the exact details are not publicly known, it is generally assumed that each search engine assigns a score to each result that satisfies a Boolean search criteria and then sorts the results according to this score. Its

[1] Interestingly, this number used to remain just below 2^{32} for at least 9 months, indicating a 32-bit hardware threshold of the previous implementation. Just on November 11^{th}, 2004 the index size jumped to about 8 thousand million pages, possibly in response to the launch of the MSN search engine.

value depends on the value of certain *features* of each webpage in the result set, e.g. its PageRank score, the text similarity between the query and the document, etc.

In this paper we approximate the underlying ranking function by analyzing query results. First, we obtain for each query result numerical values for a large number of observable features, thus converting each document into a vector. We then train our models on the difference vectors between documents at different ranks. By labeling these vectors with "+" for the direction towards the top and "–" otherwise, we reformulate our problem as a binary classification one: given a pair of documents, we try to predict which one is ranked above the other. This gives a partial order of the vectors, i.e., documents. Therefore, the binary classification trees we used do not give a complete order, i.e., a full ranking. On the other hand, for the models with linear decision boundaries, namely logistic regression and support vector machines, the normal vector to this linear boundary gives the trendline of the direction of an improvement in the underlying scores and the scalar product with this vector gives a full ranking. Although our methodology can be applied to any search engine, we chose Google as it is nowadays the most widely used [3]. More specifically, we used the Google API [5] to retrieve search results.

In section 2 we discuss previous work on this subject. We then describe our statistical methodology for estimating the ranking function in section 3. Section 4 briefly presents the components of the system. Section 5 gives our experimental results. Possible improvements and shortcomings of our approach are discussed in section 6.

2 Related Work

Pringle, Allison and Dowe [13] addressed both the problem of determining when certain pages are retrieved at all, and that of explaining how a given ranking was obtained. For the first problem they used decision trees, while the second was tackled using linear regression, exploiting the explicit scores returned by InfoSeek [8]. Both of these approaches are different from ours as is the way they obtained their data, namely by creating artificial websites and not using real web pages as we are. Furthermore, they did not use a separate test data set to obtain an unbiased estimator of the quality of their predictor.

Sedigh and Roudaki [15] presented a simple linear regression model that approximates the dynamics governing the behavior of Google where, given some observable features, they try to predict the *absolute* rank of a webpage. This has, among other things, the disadvantage that once documents are added such a prediction can no longer hold, as you trained on absolute ranks, which depend on the knowledge of all documents in the set. Our approach has the advantage that the universe of documents does not need to be known. Especially for their methodology it would have been interesting to see the performance on a separate test data set that was not used for training.

Joachims [9] presented an approach to automatically optimize the retrieval quality of search engines using clickthrough data. He rephrased the problem of learning a linear ranking as one of binary classification with linear decision boundaries and then applied Support Vector Machines (SVM) to this inference problem. We use the same reformulation, though we do not restrict ourselves to SVMs.

There is also a substantial amount of literature on various theoretical aspects of learning a ranking function, see, e.g., [2, 4]. As our focus was on obtaining an *interpretable* model for the ranking function, we limited ourselves to using logistic regression, SVMs and binary classification trees.

3 Estimating a Ranking Function

Our goal is to obtain an estimation function f for the scoring function of a search engine, and then to compare our predicted rankings with the actual rankings of this search engine.

3.1 Data set

As it is unlikely that a large search engine would employ the same ranking criterion for all types of queries, we used several sets of homogeneous queries, all dealing with a certain topic, assuming that they are ranked with the same function.

Table 1 shows our queries dataset, which is divided into four categories: Art, States, Spam and Multiple. Arts is a list of artists and States is a list of US States. For both of these categories the queries consisted of a single term. The Spam category contains phrases for which one can expect many "search engine optimized" web pages. If a search engine wants to retrieve any high quality pages for such queries it must use an elaborate

Table 1: Table of training, validation and test data classified by the query data set. Validation data is used for feature selection and tree pruning. The test data is only used to get an unbiased estimate of the generalization error.

Dataset	Type	Query terms
Arts	Training	Albertinelli, Bacchiacca, Botticelli, Botticini, Foschi, Franciabigio, Leonardo
	Validation	Michelangelo, Picasso
	Test	Pordenone, Rosselli, Verrocchio
States	Training	Arizona, Arkansas, Connecticut, Idaho, Illinois, Iowa, Kansas
	Validation	Michigan, Nevada
	Test	Ohio, Oregon, Utah
Spam	Training	buy cds, buy dvds, cheap software, download movies, download music, dvd player, free movies
	Validation	free mp3, free music
	Test	free ringtones, music videos, software downloads
Multiple	Training	anova bootstrap feature missing principal squared, analysis frequent likelihood misclassification pruning statistical, adaptive classification gating linear model proximity subset, association generalization local naive regularization test, analysis cubic gradient margin optimal risk support, automatic decision validation overfitting smoother adaptive, activation discriminant hyperplane loss single validation
	Validation	basis eigenvalue margin maximum local support, basis early information adaboost margin soft
	Test	logistic bias entropy markov piecewise loss, feature complexity gaussian logistic normal ridge training, discriminant gaussian link multiple radial supervised

ranking function. Finally, the Multiple category consists of queries with 6 terms from the domain of statistical inference.

One of our main contributions lies in the statistical rigor with which we approach the problem. Thus, for example, to obtain statistically meaningful results we partition the 12 queries for a given category into three disjoint sets:

Training Set (7 queries): We use this set of queries to learn a linear scoring function or a decision tree. The data that we actually trained on consists of difference vectors corresponding to different feature vectors. See section 3.2 for a detailed explanation.

Validation Set (2 queries): The validation set is used for greedy stepwise forward feature selection and for pruning the decision tree. In cases where features were selected directly, this set was merged with the training one. Tree pruning is explained in section 3.5.

Test Set (3 queries): To estimate the generalization error of our ranking function, we compare our predicted rankings with the observed rankings for these queries. Simply testing the performance on the training set itself leads to an overly optimistic estimate of the quality of the model. Without the use of such a set, we could have obtained wrong, but seemingly better, results due to overfitting.

3.2 Reformulation as binary classification problem

Let q be a query, and $\mathbf{u},\mathbf{v}$ be feature vector of pages, where each entry in the vector corresponds to a particular feature, e.g., how often a query term appears on the page or whether or not it appears in the URL (the list of features we considered is provided in section 4.2).

Let $\mathbf{u} <_q \mathbf{v}$ represent the ordering returned by the ranking function of a search engine for the given query, so $\mathbf{u} <_q \mathbf{v}$ if and only if the page with feature vector $\mathbf{u}$ is ranked below the one with feature vector $\mathbf{v}$ in the search engine's results to query q. In the following, we drop the subscript q.

Ultimately, we want to find a real-valued function f such that $f(\mathbf{u}) < f(\mathbf{v})$ whenever $\mathbf{u} < \mathbf{v}$. If we assume that f is linear, then there exists a vector $\mathbf{w}$ such that $f(\mathbf{u}) = \mathbf{w} \cdot \mathbf{u}$, using $\cdot$ to denote the scalar product.

Then,

$$\begin{aligned} & f(\mathbf{u}) < f(\mathbf{v}) \\ \Leftrightarrow \quad & \mathbf{w}\cdot\mathbf{u} < \mathbf{w}\cdot\mathbf{v} \\ \Leftrightarrow \quad & \mathbf{w}\cdot(\mathbf{v}-\mathbf{u}) > 0\ . \end{aligned}$$

So the problem of finding a linear function f is equivalent to finding a vector $\mathbf{w}$ such that $(\mathbf{v}-\mathbf{u})\cdot\mathbf{w} > 0$ if and only if $\mathbf{v}$ is ranked above $\mathbf{u}$. Geometrically speaking, $\mathbf{w}$ points in the direction of the increase in score value and the vectors are sorted according to the length of their projection onto this direction. If we label the difference vector $(\mathbf{v}-\mathbf{u})$ "x" if $\mathbf{v}$ is ranked above $\mathbf{u}$ and "–" otherwise we see that our problem is equivalent to finding a hyperplane with normal $\mathbf{w}$ which perfectly separates the "+" and the "–" training points.

Conversely, if we have a binary classification scheme which uses a linear decision boundary through the origin, we can use the normal vector to this plane as our $\mathbf{w}$ which defines the function f. As our data is symmetric about the origin by construction –because each comparison between two different ranks leads to two vectors pointing in opposite directions– many schemes will satisfy the property. Note that the real data will not perfectly follow a linear function in the observed features as we might not know all the underlying features which are actually used and because the function is most probably non-linear. Thus, our classification scheme needs to cope with misclassified points.

In the following we briefly introduce two classification schemes with linear decision boundaries, logistic regression models and support vector machines, and also discuss the use of binary classification trees, which are highly non-linear. See [6] for a good introduction to the area of statistical learning algorithms. We will write $\mathbf{x}$ for an either labeled or unlabeled difference vector $\mathbf{v}-\mathbf{u}$ and consider the task of classifying unlabeled points into one of the two classes "+" or "–".

It is worth noting that we not only train these linear models on the original features but we also experimented with including quadratic terms of features which, in the original feature space, leading to non-linear decision boundaries which can capture more subtle relations but are also more prone to overfitting. See section 3.7 for details on the issue of feature selection and transformations.

3.3 Logistic Regression

Logistic regression models the posterior probabilities of the classes, i.e., the probabilities of belonging to a certain class given the coordinates $\mathbf{x}$, via linear functions in the coordinate vector. In the case of only two classes "–" and "+" the model has the form

$$\log\frac{P(\text{class = "+"}|X=x)}{P(\text{class = "–"}|X=x)} = \beta_0 + \mathbf{w}\cdot\mathbf{x} \qquad (1)$$

As it can be seen easily, thisgives rise to linear decision boundaries, on which $P(\text{class = "+"}|X=x) = P(\text{class = "–"}|X=x)$. Due to the symmetry in our data the offset β_0 will be zero and the vector $\mathbf{w}$ gives the desired weights of the factors involved, indicating the direction from low ranks to high ranks. We refer the reader to [6] for a full description of the training algorithm. For our experiments we used Matlab's implementation of `glmfit` (generalized linear model) with a binomial distribution to compute the logistic regression models.

3.4 Support Vector Machines

Support Vector Machines typically use linear decision boundaries in a *transformed* feature space. The idea is that often in this higher, or even infinitely, dimensional space the data becomes separable by hyperplanes. As the mapping does not need to be computed implicitly, but only inner products defining the kernel need to be known, this approach becomes feasible. However, as we want to be able to ultimately use the related normal vector for ranking in the original feature space we only use support vector machines with linear kernels which correspond to hyperplanes in this space. For our experiments we used the `SMV light` [10] implementation with the default setting for the parameter C which is the average of $(\mathbf{x}\cdot\mathbf{x})^{-1}$. This parameter allows trading-off margin size against training error.

3.5 Binary Classification Trees

The function f that is used in practice by a real search engine might not be linear for a variety of reasons. One is that, for efficiency reasons, a search engine might use several layers of indices, where the first layer is able to answer most queries and only if this level fails the query is send to subsequent levels. Such a behavior and other simple non-linear behaviors, as using thresholds

for certain features, can at least partly be captured by decision trees.

A classification tree is built through a process known as binary recursive partitioning. The algorithm iteratively breaks up the records into two parts, examining one variable at a time and splitting the records on the basis of a dividing line in that variable. The process continues until no more useful splits can be found. Toward the end, idiosyncrasies of training records at a particular node display patterns that are peculiar only to those records. These patterns can become meaningless and often harmful for predicting unknown labels. Pruning the tree is the process of removing such leaves and branches to shrink the tree to a smaller set of crucial splits which leads to a better performance on general data. For our experiments we used Matlab's implementation of `treefit` and `treeprune`.

3.6 Selecting Difference Vectors

In most cases we trained on *all* possible pairs, which for n results are $n(n-1)$ pairs, each pair resulting in two difference vectors with opposite labels. This selection of pairs was used to train both the logistic regression models and the SVM classifiers. To reduce the computation time for the classification trees we only trained on "shifted pairs" of the form $(1, \lfloor n/2 \rfloor + 1)$, $(2, \lfloor n/2 \rfloor + 2)$, ... $(\lfloor n/2 \rfloor$,n), where again each such pair gives rise to two difference vectors. For this subset of pairs there are thus $2 \times \lceil n/2 \rceil$ points to classify. Due to the large constant difference between considered ranks these pairs are expected to be rather robust with respect to noise due to only minor differences between consecutive ranks. However, training the linear models with this subset made hardly any difference compared to training them on all pairs. So we conjecture that more pairs would not give significantly different classification trees.

3.7 Feature Selection and Transformations

The importance of the individual features for each of the four categories is given in Table 2. This table gives the precision which can be obtained on the entire data set (for each category, downloading the top 100 results per query) if we only ranked according to a single feature.

We experimented with various strategies of selecting subsets and functions of the original features as input values for the data mining algorithms. Subsets of features were selected in two ways. Firstly, in a greedy stepwise forward manner, always adding the feature which gave the best improvement on the validation data set. This strategy did not give better results than the other approaches, and thus we omit its results. Secondly, we used the p strongest features for $p \in 1, 3, 5, 10$.

Table 2: Table of the precision values obtained using only individual features to predict the ranking –feature abbreviations are explained in section 4.2. The top three features for each query set are shown in bold, and a (-) indicates a negative correlation.

Feature	Arts	States	Spam	Multiple
NBOD	(-)53.8%	**54.4%**	51.1%	50.9%
FNEN	**(-)59.4%**	53.5%	51.6%	**(-)59.8%**
RFFT	52.1%	(-)54.3%	50.0%	53.9%
ATLE	(-)54.2%	54.0%	50.0%	54.4%
FATT	56.2%	50.3%	53.0%	51.8%
AMQT	55.4%	50.5%	52.1%	56.9%
SIMT (N)	56.5%	(-)52.0%	52.7%	**59.0%**
SIMT	55.4%	(-)50.9%	52.6%	**69.7%**
TMKY (N)	51.4%	51.4%	54.5%	50.0%
TMKY	53.0%	51.4%	**55.1%**	50.0%
ILNK	**58.0%**	**66.3%**	**57.0%**	53.9%
PRNK	**58.7%**	**60.6%**	**55.0%**	57.2%
TDIR	52.3%	53.5%	54.3%	51.9%

Furthermore, we did not only use the raw features but also experimented with non-linear combinations of them. For $p \in 3, 5, 10$ we included all the quadratic terms of the selected features and trained a logistic regression model on this transformed input data. Thus, when working with features x_1, x_2 and x_3 we also include all quadratic terms such as $x_1 \times x_3$ and x_2^2. If we have p basic features then we get $p + p(p-1)/2$ new features, including the linear terms. For no data set this gave better results on the test set than working with only the untransformed linear terms.

Lastly, we tried monotone transformations such as $\log(1+x)$, $1/(1+x)$ and $1-e^{-x}$ on features such as the number of inlinks, the document length or the number of term occurrences which follow a power law. This only lead to slightly worse results which are not included here.

3.8 Normalization

Generally, the outcome of a statistical inference algorithm can depend heavily on the use of data normaliza-

tion. This is not the case in our setting. Firstly, it should be clear that multiplying the value of any feature i in all data points by a constant γ simply rescales the final scoring weight b_i by $1/\gamma$. More interestingly, also rescaling the individual difference vectors $\mathbf{x}$ has almost no influence. The sign of $\mathbf{x} \cdot \mathbf{w}$ does not depend on the length of $\mathbf{x}$, thus any linear scheme working only with the *number* of misclassified training points will yield the same result. The only property that does change is the distance from the separating hyperplane which, for instance, influences the posterior probabilities in logistic regression. However, experiments showed that the overall results were almost identical, also for the classification trees, with very slightly better results obtained by training on the unnormalized difference vectors. All experimental numbers refer to this unnormalized setting.

4 System Architecture

To extract the feature sets used in our analysis we built a system with three components: a downloader, a parser, and an analyzer.

Downloader: software that executes a query and downloads the returned pages using the data set queries

Feature Extractor: software that computes the features of the pages downloaded.

Analyzer: software that analyzes the features of the returned pages and estimates a function using numerical methods.

4.1 Downloader

The downloader receives as an input a data set of queries (see Table 1). For each such query it does the following:

1. Submits the query to the search engine, i.e. to the Google API in our case, downloads all URLs u_i obtained as result set, and checks whether they or their domain are contained in the Open Directory [12]. This is another feature which might influence the ranking function of a search engine, and thus we included it in our study. For simplicity and to avoid bias based on the file type, we limited ourselves to HTML pages.
2. For each u_i, it then sends a query to obtain the number of inlinks, using Google API's "link:www.abc.om" syntax, as well as the top 5 in-linking pages. We had to limit ourselves to 5 inlinks because of the reduced access offered by the Google API to the search service, i.e., only 1000 queries of 10 answers per day, per user.
3. Finally, it downloads these 5 pages to further analyze their anchor text.

Also, for each individual query term, the "Downloader" submits a single query and outputs the estimate number of results. This will be later used to compute the inverse document frequency value for the tf-idf similarities [14].

4.2 Feature Extractor

The Feature Extractor receives as an input the downloaded pages. Pages are converted to XML using `tidy`, to be able to use the DOM (document object model) API for accessing different parts of the document. This is useful for checking if query terms appear in some special formatting element such as boldface.

We divide the list of the page features that are extracted by the parser into: content, formatting, link and metadata. Where sensible we included both raw and normalized versions of features, e.g. counting both the absolute number of query term occurrences in the title and the percentage of query terms appearing there. Here is the complete list of features we extracted. An (N) after a feature indicates that we also considered a normalized/averaged version of this feature.

Content features, query independent:

- `DIFT` Number of different terms of the document
- `FNEN` Fraction of terms in the documents which can not be found in an English dictionary
- `NBOD` Number of bytes of the original document
- `NBTO` Number of bytes of text in the original document
- `RFFT` Relative frequency of the more frequent term, i.e.: term frequency. If a document has 3 words and the most frequent word repeats 2 times, then this is 2/3
- `ATLE` Average term length

Content features, query dependent:

- TFQT Term frequency of query term = Number of occurrences of the query term (averaged over the different query terms) (N)
- SIMT Similarity of the term to the document, in terms of vector space model. We compute it using the frequency of terms in documents and the inverse document frequency of each term. (N)
- APQT Average position of the query terms in the document = 1 at the beginning, 0 if at the end and in-between in the middle.
- AMQT Average matches of the query terms
- CTQW Closeness of terms in the query in the webpage (distance in number of terms , smallest windows containing all of them)
- FATT Anchor text term frequency

Formatting features, query-dependent. We used the number of "special" occurrences divided by the total number of occurrences of query terms:

- THTM Term in a special document zone including HTML tags: B, I, U, FONT, BIG, H1–H6, A, LI and TITLE (N)
- TATV Term as an attribute value (element/@attribute): IMG/@ALT, IMG/@TITLE (N)
- TMKY Term in the meta keywords or description (N)
- TCLP Term in capitals in the page (N)

Link features, query-independent:

- ILNK Number of pages linking to a page, in-degree approximated using Google API link: queries
- PRNK PageRank of the page, or the approximation of the PageRank in a 0-10 scale obtained from Google's toolbar.
- OLNK Number of out-links in the page
- FOLN Fraction of out-links to external Web sites

Metadata features:

- TURL Term is in the page's URL or not.
- TDIR Term is listed in a web directory or not.

We computed text similarity SIMT of the query and the returned document. SIMT is computed using a TF-IDF weighting scheme similar to the one used by Salton [14], where the similarity is defined using the weight of each query term. This weight is computed using the normalized frequency of terms in documents and the idf inverse document frequency of each term.

4.3 Analyzer

The analyzer is the component is the last stage of our system and obtains the estimate of the ranking function. The statistical methods used to obtain this estimate were discussed in section 3. To evaluate the performance we used the following quality measures, each having its own justification.

1. **Precision on all pairs:** This measure simply looks at *all* possible pairs and the corresponding difference vectors. The precision is the percentage of correctly classified vectors, thus corresponding to a correct "**u** is ranked above **v**" decision. For the cases where we also have a total ordering, namely for logistic regression and the SVM model, this measure can be computed from the Kendall's τ measure as: 50%+ 50 * Kendall's τ %.
2. **Precision on "shifted" pairs:** See 3.6 for a description of the "shifted" pairs. Here we only look at the percentage of correctly classified "shifted" pairs, which are further apart in the original ranking and their relative order is thus easier to predict.
3. **Precision on "top" pairs:** Here we only consider for each query the top result and all its difference vectors. This number thus gives the percentage of web pages which are (correctly) predicted to be ranked below the highest ranking document.

5 Experimental Results

To give a quantitative estimate of the importance of each individual feature Table 2 gives the precision values which can be obtained for each category if one ranked *only* according to this individual feature. However, only features which were among the top 5 strongest for at least one category are included in the table. A (-) indicates that the feature is negatively correlated with the score, i.e., an increase in the feature value is an indicator for a worse ranking. Note that using only the strongest feature gives a precision between

Table 3: Ranking for the query ”discriminant gaussian link multiple radial supervised” from the Multiple query set when using *only* the SIMT feature. There were 33 results for this query.

Predicted rank	1	2	3	4	5	6
Google rank	3	5	6	4	13	2

57% for the Spam category and 69.7% for the Multiple category. Table 3 gives the top 6 results of a query ranking obtained when only using the text similarity (SIMT) feature. For the Arts category the strongest indicator of a high ranking was a low fraction of non-English terms (FNEN), followed by the PageRank (PRNK). For the States queries the by far strongest such indicator was the number of in-links (ILNK). For the Spam query this was also the strongest feature but far less significantly, emphasizing that for these queries it would be fatal to put too much weight on any individual feature. Out of all four categories it was also this category where the directory information (TDIR) was most closely linked to the ranking. Lastly, for the long queries from the Multiple category the SIMT was most closely correlated with the ranking.

The best performances of any model for each category are listed in Table 4. Unfortunately, these are only marginally better than the baseline values for the strongest feature in Table 2. Table 4 also lists the details of the corresponding model. The best pruned trees consisted in almost all cases of a single node corresponding to the strongest individual feature. The fact that the SVMs did not give an improvement over the baseline for any model might be due to an inappropriate (default) choice of the regularization parameter C with which the authors did not experiment.

It is worthing pointing out that the precision is significantly higher for the “shifted” pairs, as can be seen from the second column of Table 4. These pairs are further apart and are thus easier to classify as, in general, the differences in the relevant features will be more striking.

6 Shortcomings and Room for Improvement

As across all data sets and for all methods and feature transformations considered the best test precision was only about 65%, as shown in Table 4 the question arises, how this could be improved?.

Working with only our collection of features we can, given the number of different methods and transformations we tried, safely claim that the answer is a negative one: it cannot be improved substantially. More features, such as the domaing ending (.edu, .com, etc.), which also could have an influence on the ranking, could be included in the analysis but are unlikely to give a dramatic boost. Similarly, much larger training data sets would probably exhibit only a minor influence; in our case we were strongly hindered by the query limitations of the Google API. The real problem seems to lie in the fact that many crucial features are hidden and cannot be observed from the outside.

These features which are certainly relevant may include:

- The query logs, which Google obtains through its toolbar.
- The age of the incoming links and other information related to web link dynamics.
- The rate of change at which a website changes, obtained by repeated web crawls.
- The “true” number of ingoing links, as Google’s `link:www.abc.com` only gives a lower bound.
- The “true” PageRank used by Google, as the one displayed in its toolbar is only an approximation, and furthermore, seems to be too strongly correlated to the number of in-links [16].

Some of these could, however, theoretically be obtained by a web search engine with a large enough set of indexed web pages. It might also be worth including a category with random, artificial terms or numbers, assuming that there are still a few hits for these terms. For such a category at least the use of query logs could be largely ruled out.

More fundamentally, one can only speculate about the algorithmic details. It is, e.g., possible that Google uses (variants of) topic-sensitive PageRank [7] or the Hilltop algorithm [1], both of which try to overcome the

Table 4: Best precision achieved on all, "shifted" and "top" pairs. We include the performance on the test data as well as on the whole data set, including training, validation and test sets.

	% all pairs correct		% "shifted" pairs correct		% "top" pairs correct		
Dataset	Test	All	Test	All	Test	All	Best model
Arts	63.7%	61.8%	69.1%	66.4%	47.6%	48.0%	Log. regr., strongest 3 features
States	64.6%	66.3%	73.2%	73.8%	97.6%	98.5%	Class. tree, only ILINK feature
Spam	62.5%	59.5%	70.5%	62.1%	98.2%	74.8%	Log. regr., strongest 10 features
Multiple	67.5%	70.9%	78.1%	81.3%	81.0%	87.0%	Log. reg., strongest 3 features

notion of a global, topic-independent measure of quality, which is inherent to PageRank. For these algorithms the ranking would no longer be a function of simple features and a much more elaborate analysis would be needed.

One should also not forget that any web search engine always has the option of "manually" re-ranking the results for certain queries. It is known that in certain countries search engines voluntarily cooperate with the authorities to exclude certain web pages for legal reasons from their results. Likewise, it is imaginable that for certain queries pages are pushed up or down because of financial or other agreements.

The reason for us to choose different query categories and to try to have "homogeneous" queries within one category was that we assume that a query is first categorized and then handled according to the categorization. This categorization could involve the scan for certain keywords indicating a certain topic but it could also involve inferring information about the type of question (homepage finding vs. question answering) and the type of user. A query such as "I'm looking for information about search engines" containing several stopwords, might indicate a user less familiar with using search engines and thus less careful in choosing the query terms. This could imply a boost of query-independent features for such queries. Likewise, a user using advanced query syntax who "knows what he is doing" might be better off with a different ranking scheme. This could be the reason why the queries "adversarial" and "adversarial -lairasrevda" (excluding adversarial spelled backwards) lead to different rankings on Google, although conceptually there is no reason for this.

It is even possible –and sensible– for a web search engine to take information about the query initiator into account. Such information could either be collected in the form of data cookies or, simply, by considering the browser language, connection type or geographical location. A user with a dial-up connection will generally have a different user profile from a user of a high-speed university domain so that different ranking schemes might be appropriate. Similarly, a user in Spain might prefer a different ranking than a user in Germany.

7 Conclusions

Along this study we attempted to produce a complete method for learning search engine ranking function(s). Overall, our experiment was sound and its results were very consistent: ranking only according to the strongest feature for a category gives is able to predict the order in which any pair of pages will appear in the results with a precision of between 57% (for a data set including commercial terms that are used in spam) and 70% (for a data set including long queries from a very specific domain). This was, despite trying various algorithms and feature transformations, only mildly improved by including other features.

If one had not split the data into a training and a test set one could have, given the large number of features and transformations we considered, achieved an almost arbitrarily high precision on the training data, but a worse performance on unseen data. In this article we have also discussed the reasons for these results, which are likely to be related to the lack of many crucial features such as user preferences data and algorithmic details such as the possible use of topic-sensitive PageRank.

Acknowledgment

The authors would like to thank Dr. Jörg Rahnenführer for useful suggestions and clarifications concerning some aspects of the data mining algorithms used.

References

[1] K. Bharat and G. A. Mihaila. When experts agree: using non-affiliated experts to rank popular topics. In *WWW '01: Proceedings of the tenth international conference on World Wide Web*, pages 597–602, 2001.

[2] W. Cohen, R. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.

[3] comScore Networks. Press release, 2004.

[4] J. Díez, J. J. del Coz, O. Luaces, F. Goyache, A. M. P. J. Alonso, and A. Bahamonde. Learning to assess from pair-wise comparisons. *LNCS*, 2527, 2002.

[5] Google api. http://api.google.com.

[6] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2003.

[7] T. H. Haveliwala. Topic-sensitive pagerank. In *WWW '02: Proceedings of the eleventh international conference on World Wide Web*, pages 517–526. ACM Press, 2002.

[8] Infoseek search engine. http://www.infoseek.co.uk/.

[9] T. Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM Press, 2002.

[10] T. Joachims. Svm-light support vector machine, 2002. http://svmlight.joachims.org/.

[11] J. Nielsen. When search engines become answer engines. *Jakob Nielsen's Alertbox*, August 2004.

[12] Open directory project. http://dmoz.org/.

[13] G. Pringle, L. Allison, and D. L. Dowe. What is a tall poppy among web pages? *Computer Networks and ISDN Systems*, 30:369–377, 1998.

[14] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[15] A. K. Sedigh and M. Roudaki. Identification of the dynamics of the google's ranking algorithm. In *13th IFAC Symposium On System Identification*, 2003.

[16] T. Upstill, N. Craswell, and D. Hawking. Predicting fame and fortune: Pagerank or indegree? In *Proceedings of the Australasian Document Computing Symposium, ADCS2003*, pages 31–40, Canberra, Australia, December 2003

Optimal Link Bombs are Uncoordinated *

Sibel Adalı, Tina Liu, Malik Magdon-Ismail
Department of Computer Science, Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180.
Email: {sibel, liut2, magdon}@cs.rpi.edu

Abstract

We analyze the recent phenomenon termed a *Link Bomb*, and investigate the optimal attack pattern for a group of web pages attempting to link bomb a specific web page. The typical *modus operandi* of a link bomb is to associate a particular page with a search text and then boost that page's *pagerank*. (The attacking pages can only control their own content and outgoing links.) Thus, when a search is initiated with the text, a high prominence will be given to the attacked page. We show that the best organization of links among the attacking group to maximize the increase in rank of the attacked node is the *direct individual attack*, where every attacker points directly to the victim and nowhere else. We also discuss optimal attack patterns for a group that wants to hide itself by not pointing directly to the victim. We quantify our results with experiments on a variety of random graph models.

1 Introduction

Generally, a search on a particular topic on a particular search engine (such as *Google*) will output a ranked list of relevent web pages. The prominence of a page in this listing is an important indicator of how many people will visit the page. For a commercial web site, its prominence with respect to product searches has important financial consequences, as does the prominence of a competitor's website with respect to slander about products. Prominence in rankings is prestigious and can add credibility [10].

As a result of the importance attached to one's pagerank, especially one's Google pagerank, artificial methods for boosting one's pagerank with respect to a particular topic have become an active area for discussion. A prominent case was an attempt to "bring down the White house" by giving high prominence (in fact the primo ranking) to a web-biography of the U.S. President with respect to the text "miserable failure" [10, 14]. Such attacks are generally termed *Google bombs* (named after the success of such attacks on Google rankings), which are attempts to give prominence to a particular web page with respect to a particular (usually derogatory) piece of text[1]. We study *link bombs* which attempt to alter the rankings obtained through the PageRank algorithm by manipulating the links - such attacks would be relevant to any search engine that uses such a page ranking algorithm. Link bombs need not be derogatory, for example, a web-retailer could also make use of link

*This work was partially supported by the National Science Foundation under grant EIA-0091505

[1] After the first attack (which was with respect to the text "talentless hack"), several other attacks also succeeded in raising the ranks of web pages with respect to specific keyword(s), in some cases using as few as 25 links. It has been argued that several factors contribute to the success of an attack, eg. the number and prominence of the attacking pages; the (un)popularity of the keyword. Many of these attacks were usually initiated by Blogs which tend to be updated often and have a lot of content. It has been argued that these factors contribute to the high prominence of Blogs which in turn have higher influence in the pagerank of other pages. Similarly, some of the keywords chosen were very rare in the web pages, such as "French Military Victories". However, even attacks using keywords as popular as "Weapons of Mass Destruction" have been successful (BBC News, Sunday, 7 December, 2003).

bombs to improve the prominence of its own website with respect to a particular topic(s). The link bombers are usually some (coordinated) set of web pages which add outgoing links to their web page. Some of these links will point to the attacked page, and contain the text they (the bombers) are trying to associate with the attacked page. The issue we address is how these bombers should organize their outgoing links in order to maximize the success of their link bomb.

There is currently a great deal of discussion on whether a link bomb can be considered an "undesirable" attack [14] that exploits a weakness in the pagerank algorithm [6, 12]. The pagerank algorithm assigns you a pagerank by considering the number and importance (according to PageRank) of web pages that point to you. Given that a search engine like Google currently ranks about 8 billion pages, one would expect that a very small number of web pages should not be able to change the ranking of a page dramatically, contrary to what has been observed. Thus, one motivation for studying the optimal attack is to determine specific abnormal but effective attack patterns that could be identified as artificial Google bombs.

We present results on the optimal link bomb. Specifically, the *attackers* are a set of web pages whose outgoing links can be manipulated, and the *victim* is the target web page to be bombed. Our main result is to establish the following theorem as a starting point for a discussion of accountability on linked structures such as the WWW,

Theorem. *The attack which maximizes the* **rank** *of the victim with respect to page rank is the direct individual attack.*

The *direct individual attack* is the attack in which every attacker points *only* to the victim and to no other page. In particular, in the optimal attack, none of the attackers point to each other. Thus, the optimal attack masquerades as a set of uncoordinated "random" nodes, all pointing to the same page. We also discuss optimal "disguised" attack patterns, in which none of the attackers wish to directly point to the victim – all paths from the attackers to the victim must be of at least some minimum length. In this case the optimal attack is still a direct individual attack, however now the attackers point to some other *intermediate node* (not the victim).

While the optimal attack is always the direct individual attack, the amount by which the direct individual attack surpasses other (more coordinated) attack patterns may depend on the nature of the graph. We give experimental results that quantify this phenomenon for a variety of different attack patterns. On certain random graph models of the Web, some coordinated attack patterns are almost as good as the direct individual attack, and can hence be used in place of the direct individual attack as a means of disguising the attack. While the effect of graph structure on the pagerank has been investigated in the literature [11, 6], to our knowledge, these are the first results regarding the effect of the graph structure on the effectiveness of link bombs.

Our results raise interesting questions such as how to detect and respond to link bomb attacks (in general this problem is NP-hard, see for example [16]). Since the attackers will have no visible associations amongst themselves, it is hard to detect and prove that they are participating in an attack. If the optimal attack were a tree structure, there would be a small set of nodes with high prominence that one might argue are "responsible" for the attack. The other nodes pointing to these nodes could also be held accountable aiding and abbeting the actions of the responsible nodes. Such accountability is not possible in an individual attack.

We proceed by first discussing some preliminary definitions, followed by a preview of our result for an isolated graph, in which the only nodes are the attackers and the victim. We then discuss general graphs, followed by some experimental results on a variety of random graph models. We conclude with a discussion of the implications of our results. (We omit technical proofs which can be found in a full version of this paper [1].)

2 Preliminaries

A search query on a set of keywords results in an ordered list of web pages $\mathcal{W} = \{\omega_i\}$. Each web page $\omega \in \mathcal{W}$ contains some or all of the keywords either in its text or in the text of a link that points from some other web page to ω. A scoring function is used to order the pages in $\mathcal{W}$. The most prominent page (page with the highest score) is given rank 1, etc.

Google [3] considers many factors in its scoring function, including: keyword frequency; relative locations of the keywords; the position and style of the keywords. An important factor in the scoring function is the *pagerank* which depends on how the web page is embedded in the entire graph of web pages. An early paper on the Google system [3] suggests that no one factor dominates the scoring function, however, the pagerank plays an important role. In this paper, we will concentrate only on the pagerank factor and discuss how it can be manipulated.

The *web graph* is a directed graph $G = (V, E)$ that models the World Wide Web. The vertex set V represents the pages and documents, and the edge set E represents the links between the pages and documents[2]. The edges are directed: if $(v_1, v_2) \in E$, then v_1 contains a link to v_2. In a web graph, the in-degree $indeg(v)$ of page v is the number of links that point to v and the out-degree $outdeg(v)$ is the number of links originating from v that point to other pages. A (directed) *path* of length ℓ is a sequence of vertices $v_0, v_1, \ldots, v_\ell$ with $(v_{i-1}, v_i) \in E$ for $i = 1, \ldots, \ell$. v_ℓ is the terminal node in the path, and $v_1, \ldots, v_{\ell-1}$ are intermediate nodes. We allow parallel edges between two vertices, but no self-loops.

The pagerank p_i models the probability that node i will be visited either by randomly navigating down links in the web graph or by randomly jumping to page i. Let α be the probability to navigate, and $1 - \alpha$ the probability to jump. Then the pageranks $\{p_j\}$ of the nodes in a graph simultaneously satisfy the set of linear equations[3]

$$p_i = \alpha \sum_{(v_j, v_i) \in E} \frac{p_j}{outdeg(v_j)} + \frac{1-\alpha}{N}. \qquad (1)$$

($0 \le \alpha \le 1$ and $N = |V|$.) The first term represents the probability to reach i by random navigation. An edge may appear multiple times if there are parallel links. The second term represents the probability to reach i by randomly jumping. Typically, $\alpha \in [0.85, 0.95]$. p_i is larger if v_i has a large in-degree, and its incoming links are from high pagerank nodes with small out-degree. The PageRank algorithm [12] is an iterative approach to solving these equations. The pageranks are all initialized to $p_i^0 = \frac{1}{N}$. The PageRank iteration [12] is given by

$$p_i^{t+1} = \alpha \sum_{(v_j, v_i) \in E} \frac{p_j^t}{outdeg(v_j)} + \frac{1-\alpha}{N}. \qquad (2)$$

p_i^t converges to the (unique) solution of (1). Every page can manipulate its outgoing links, but it cannot change its incoming links.

A *link bomb*, or *attack* occurs when a group of *attackers* $A = \{v_1, \ldots, v_K\}$ alters their outgoing links so as to boost the pagerank of a *victim* $v_0 \notin A$. Before the attack, if the edge set is E, then after the attack the edge set will be $\bar{E}$ where the only edges added or removed from E are of the form (v_i, u) where $1 \le i \le K$ and $u \in V$, i.e., the attackers may remove and/or add outgoing links only. After the attack, the new web graph is $\bar{G} = (V, \bar{E})$. Let p_i denote the pageranks in the original graph G (before the attack), and $\bar{p}_i$ the pageranks in $\bar{G}$ (after the attack). The *magnitude* of the attack $\Delta p_0 = \bar{p}_0 - p_0$ is the amount by which the pagerank of the victim increased, and is a measure of the success of the attack. In our analysis, we only consider the magnitude of the attack,

[2]Note that the definition of an edge is traditionally given by hyperlinks in a web page. However, it is also possible to count URLs in the body of a web page as links. The definition of what constitutes a link is usually application dependent.

[3]An alternative and common formulation of the pageranks in the literature is as the stationary distribution of a suitably defined finite irreducible Markov chain with transition matrix $P = (1-\alpha)M + \alpha U$, where U is a matrix of 1's. Many of our results could be obtained by analyzing how the stationary distribution changes under perturbations of P. Our approach is more graph theoretic, treating the problem as a flow.

and assume that all other factors entering into the scoring function are unchanged.

3 The Optimal Link Bomb

In this section, we investigate how to maximize the magnitude of the attack. In particular, we show that the effectiveness of the attack *does not* increase if the attackers try to coordinate the attack in some way, by introducing links among themselves in order to increase their ranks. (Recall that, incoming links from higher ranked pages are more beneficial to your rank.) First, we consider a simplified case, in which the attackers and the victim are isolated from the rest of the graph. We then consider the general case.

3.1 Isolated Graphs

We can restrict our attention to the vertex set composed of the attackers and the victim, $V = A \cup v_0$ (i.e., $N = |V| = K + 1$). Assume (for simplicity) that v_0 does not point to any member of A. We first consider some examples of attacks, before giving the general result. In all cases, all the attackers A point to the victim v_0, and what differentiates the attacks is how the attackers are themselves organized.

Direct Individual: The only links are to v_0.

Tree: The attackers form a tree. For any graph with a topological order, one can compute the page ranks efficiently (in linear time). For analysis purposes, we will specialize to a *star attack* in which $v_2 \ldots, v_K$ point to v_1 and all attackers point to v_0.

Cycle: The attackers form a cycle.

Complete: The attackers a complete graph.

By solving the linear system (1) for the graph resulting from each of these attacks, we obtain

Lemma 1 *For the isolated graph,*

$$\begin{aligned}
\bar{p}_0(\mathit{individual}) &= p_0(1+\alpha K), \\
\bar{p}_0(\mathit{star}) &= p_0\left(1+\tfrac{\alpha}{2}(K(1+\alpha)+1-\alpha)\right), \\
\bar{p}_0(\mathit{cycle}) &= p_0\left(1+\tfrac{\alpha K}{2-\alpha}\right), \\
\bar{p}_0(\mathit{complete}) &= p_0\left(1+\tfrac{\alpha K}{K(1-\alpha)+\alpha}\right),
\end{aligned}$$

where $p_0 = (1-\alpha)/(K+1)$.

Since $0 \le \alpha \le 1$, after some algebra, we obtain

Theorem 1 *For the isolated graph,*

$$\bar{p}_0(\mathit{individual}) \ge \bar{p}_0(\mathit{star}) \ge \bar{p}_0(\mathit{cycle}) \ge \bar{p}_0(\mathit{complete}).$$

In fact, the direct individual attack is optimal for the isolated graph:

Theorem 2 *For an isolated graph, p_0 is maximized (uniquely) by the individual attack.*

3.2 Arbitrary Graphs

When $v_0, \ldots, v_K$ are embedded in a larger graph G, the direct individual attack is still optimal. Intuitively, one can view the PageRank iteration (2) as sending a flow of pagerank down the directed edges. The maximum flow from v_i to v_0 occurs when v_i points directly to v_0, and to no other node – any other links divert the flow and lead to a lower magnitude attack. The following results will make this intuition more formal. We will generally refer to nodes which are neither the attackers nor the victim by w_j, and u_j will be used to refer to any node. The 1-neighborhood $N_1(v)$ of a node v is the set of nodes to which v points. $N_k(v)$ $(k > 1)$ is the set of k-neighborhood nodes: $u \in N_k(v)$ *iff* for some $w \in N_{k-1}(v)$, $(w,u) \in E$. Note that v could be in its own k-neighborhood for $k > 1$, and $N_0(v) = \{v\}$.

Consider attacker v_i, and, without loss of generality, assume it initially has no outgoing links. Suppose now that it adds δ outgoing edges. This results in $\frac{\alpha}{\delta}$ of its rank "flowing" along each of its edges to its

neighbors (note there may be parallel links). Thus, the rank increase for a 1-neighbor u_j is given by

$$\Delta_j^1 = \alpha \sum_{(v_i,u_j)\in E} \frac{p_i}{outdeg(v_i)},$$

where the superscript 1 indicates that u_j is a 1-neighbor, and j is an index that enumerates the 1-neighbors. The sum is over all parallel edges that v_i may have to u_j. This increase in rank in turn propagates to 2-neighbors, resulting in an increase in the rank of a 2-neighbor u_k by an amount

$$\Delta_k^2 = \alpha \sum_{\substack{(u_j,u_k)\in E \\ s.t\ u_j\in N_1(v_i)}} \frac{\Delta_j^1}{outdeg(u_j)}.$$

The sum is over all 1-neighbors pointing to u_k (including parallel edges). If the newly added edges create a path from v_i to v_0, then some amount of v_i's pagerank will propagate to v_0. We define Δ_j^l to be the change in the page rank of u_j from flow down all paths of length l from v_i to u_j,

$$\Delta_j^l = \alpha \sum_{\substack{(u_k,u_j)\in E \\ s.t\ u_k\in N_{l-1}(v_i)}} \frac{\Delta_k^{l-1}}{outdeg(u_k)}$$

Let $\delta(l)$ be total increase in page rank through paths of length l, $\delta(l) = \sum_j \Delta_j^l$. Since the pagerank increase attenuates by a factor α with each edge, we have the following lemma.

Lemma 2 *$\delta(l) \le \alpha^l p_i$, with equality* iff *$\delta(l-1) = \alpha^{l-1} p_i$ and for every $u_k \in N_{l-1}(v_i)$, $outdeg(u_k) > 0$.*

Let $\mathcal{S}$ be a set of nodes. A path q passes through $\mathcal{S}$ if some node of $\mathcal{S}$ is an intermediate node of q. A set of paths P pass through $\mathcal{S}$ if every path in P passes through $\mathcal{S}$. Let P_t be a collection of paths that passes through $\mathcal{S}$, with every path in P_t having the same terminal node $t \neq v_i$ (t is not an intermediate node of any path in P_t). We call t a *progeny* of $\mathcal{S}$ with respect to the paths P_t. Since every path passes through $\mathcal{S}$, some prefix of every path in P_t has a terminal node in $\mathcal{S}$. For each path $q \in P_t$, let $q_{\mathcal{S}}$ be a (any) prefix with terminal node in $\mathcal{S}$, and let $P_t(\mathcal{S})$ denote the collection of such distinct prefixes $\{q_{\mathcal{S}}\}$.

The *influence* $I(\mathcal{S}|P_t(\mathcal{S}))$ of v_i on $\mathcal{S}$ is the total flow of pagerank (summed over all nodes in $\mathcal{S}$) from v_i to $\mathcal{S}$ along the paths in $P_t(\mathcal{S})$ (which are (distinct) prefixes in P_t). The influence $I(t|P_t)$ of v_i on t is the total flow of pagerank that flows to t along the paths in P_t (which pass through $\mathcal{S}$). Every path in P_t has at least one additional edge compared with its corresponding prefix that terminates in $\mathcal{S}$, so the influence that propagates to t along P_t can be at most the influence that propagates to $\mathcal{S}$ along the paths in $P_t(\mathcal{S})$, attenuated by a factor α. We have the following lemma.

Lemma 3 *$I(t|P_t) \le \alpha I(\mathcal{S}|P_t(\mathcal{S}))$, independent of which prefixes are used in the construction of $P_t(\mathcal{S})$.*

We now consider v_i's attack on v_0. Let P denote the collection of all (distinct) paths from v_i to v_0 in which v_0 appears only as the terminal node, i.e., v_0 is not an intermediate node of any path in P. Note that if there are cycles in the graph, then P may contain an infinite number of paths. Let the flow of pagerank from v_i to v_0 down the paths in P be denoted Δ. There may be cycles containing v_0, in which case, the pagerank increase Δ will continue to flow around these cycles, back to v_0 increasing the pagerank further, i.e., Δ will be amplified by the cycles. Let Δp_0^i be v_i's contribution to the magnitude of the attack,

$$\Delta p_0^i(\Delta) = \Delta + \mathrm{amp}(\Delta),$$

where $\mathrm{amp}(\Delta)$ is the amplification due to the cycles that contain v_0. The larger Δ, the larger will be the amplification of Δ,

Lemma 4 *$\Delta p_0^i(\Delta)$ is monotonically increasing.*

Lemmas 2, 3 and 4 are the main tools we will need to prove our main result, namely that the individual attack is optimal. By Lemma 4, since Δp_0^i is monotonically increasing in Δ, Δp_0^i will be maximized when Δ is maximized. Δ is given by the sum of the flows of pagerank from v_i to v_0 along the paths in P, therefore we only need to consider this flow.

Let ℓ be the length of the shortest path in P (there may be many such shortest paths). Consider the set L of all distinct paths of length ℓ originating at v_i. Some of these paths have terminal node v_0. We now restrict our attention to the set L' containing those paths in L which do not have terminal node v_0. Note that none of the paths in L' can have v_0 as an intermediate node since the shortest path from v_i to v_0 has length ℓ. Let Let $\mathcal{S}$ denote the set of terminal nodes in L'. Partition P into two disjoint sets, P_ℓ and $P_{>\ell}$, where P_ℓ contains the paths in P with length ℓ and $P_{>\ell}$ the paths with length $> \ell$. Every path in $P_{>\ell}$ must pass through at least one of the nodes in $\mathcal{S}$, therefore $P_{>\ell}$ passes through $\mathcal{S}$. Every path in $P_{>\ell}$ has terminal node v_0, and v_0 does not appear as an intermediate node in any of these paths. Thus, v_0 is a progeny of $\mathcal{S}$ with respect to $P_{>\ell}$. Every path in $P_{>\ell}$ has a prefix of length ℓ with terminal node in $\mathcal{S}$. Collect these distinct prefixes into the set $P_{>\ell}(\mathcal{S})$.

Let Δ_ℓ be the contribution to Δ due to flow along the paths in P_ℓ, and $\Delta_{>\ell}$ the contribution due to flow along the paths in $P_{>\ell}$. Then,

$$\begin{aligned}
\Delta &= \Delta_\ell + \Delta_{>\ell} \overset{(a)}{=} \Delta^\ell_{v_0} + I(v_0|P_{>\ell}),\\
&\overset{(b)}{\leq} \Delta^\ell_{v_0} + \alpha I(\mathcal{S}|P_{>\ell}(\mathcal{S})),\\
&\overset{(c)}{\leq} \Delta^\ell_{v_0} + I(\mathcal{S}|P_{>\ell}(\mathcal{S})),\\
&\overset{(d)}{\leq} \Delta^\ell_{v_0} + \sum_{s\in\mathcal{S}} \Delta^\ell_s \overset{(e)}{=} \delta(\ell) \overset{(f)}{\leq} \alpha^\ell p_i.
\end{aligned}$$

(a) follows from the definitions of $\Delta^\ell_{v_0}$ and influence; (b) follows from Lemma 3 and (c) because $\alpha \leq 1$. (d) follows because the paths in $P_{>\ell}(\mathcal{S})$ are all of length ℓ, so $P_{>\ell}(\mathcal{S})$ is a subset of all the paths of length ℓ that terminate in $\mathcal{S}$; (e) follows from the definition of $\delta(\ell)$, since $\mathcal{S} \cup v_0 = N_\ell(v_i)$; finally, (f) is an application of Lemma 2. Equality occurs *iff* $\mathcal{S}$ is empty, and all paths from v_i are of length ℓ, ending at v_0. Certainly, the optimal value of ℓ is 1, and so we have the following theorem[4].

[4]An alternative proof of this theorem using the Markov chain approach can be given using a generalization of the result in [5], where it is shown that adding the edge (i, j) can only increase the rank of j.

Theorem 3 *Δp^i_0 is maximized if and only if the* only *edge from v_i is to v_0. This is independent of all the other edges in the graph, in particular independent of the edges from the other v_j.*

Theorem 3 directly implies the following result,

Corollary 1 *The direct individual attack is optimal.*

A related issue is whether the direct individual attack also maximizes the rank (as opposed to the pagerank) of the victim. This question is not immediately answered by Theorem 3 since the actual rank depends on the *relative* pagerank of v_0 with respect to the other nodes, and not the absolute pagerank of v_0. We will now show that the rank is also maximized by the direct individual attack.

Suppose that some other attack X maximizes the rank of v_0. This means that for some node u, $\bar{p}^I_{v_0} \leq \bar{p}^I_u$ and $\bar{p}^X_{v_0} > \bar{p}^X_u$ (I denotes the direct individual attack). We show that such a situation can never occur, leading to the following result.

Theorem 4 *The direct individual attack maximizes the rank of v_0.*

3.3 The Optimal Disguised Attack

We now consider the situation in which the attackers wish to maximize the magnitude of their attack on v_0, but they wish to disguise the attack by not pointing directly to the victim. In such an attack, the anchor text will be associated to the victim, hence we assume that the victim already has a high prominence with respect to the anchor text. The specific disguise constraint we consider is that for every attacker, the shortest path to the victim should have length at least $\ell \geq 1$.

Consider attacker v_i. In any attack, some amount of pagerank flows from v_i to v_0. In any directed graph, we define $f(u; v)$, the *forward value* of vertex u with respect to vertex v, to be the fraction of u's pagerank that flows to v along paths with v as

terminal node but not as intermediate node. Thus, for example, $f(v;v) = 1$. Since the fraction of u's rank that makes it to v can be obtained by multiplying the fraction flowing to each neighbor with the fraction flowing from that neighbor to v, we obtain the *forward equation* for the forward values $f(u;v)$:

$$\begin{aligned} f(v;v) &= 1, \\ f(u;v) &= \frac{\alpha}{outdeg(u)} \sum_{(u,w)\in E} f(w;v). \end{aligned} \quad (3)$$

The forward equation (3) is similar to the pagerank equation (1) and can be solved by an iterative algorithm similar to the PageRank iteration [12].

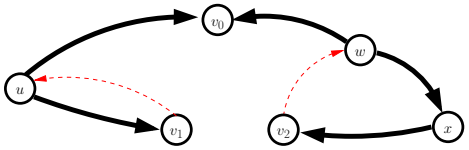
For every vertex u (not an attacker), we consider the edge set $E_u = E \cup (v_i, u)$, which defines a new directed graph in which the edge set is augmented by a single link from the attacker to u. For this graph, we can compute the forward value $f_u(w; v_0)$ of any vertex w with respect to v_0. We define the value $V_i(u)$ of vertex u to attacker v_i by

$$V_i(u) = f_u(v_i; v_0).$$

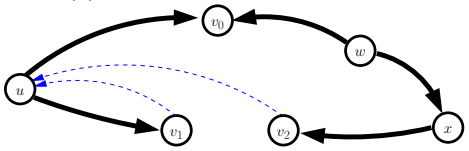
By Lemma 4 the optimal attack is the one that maximizes the flow of pagerank to v_0, which means that v_i should point to the node u satisfying the "disguise constraints" that maximizes $V_i(u)$. Arguments similar to those that led to Theorem 3 give

Theorem 5 *The optimal disguised attack for a single attacker v_i is a* single *link to the vertex u, at distance $\ell - 1$ from v_0, which maximizes $V_i(u)$.*

Unfortunately, the maximizing node $V_i(u)$ need not be the same for different attackers – the disguise constraint introduces dependencies between attackers, i.e., the optimal attack for a particular attacker may depend on what the other attackers do. In particular, it is no longer the case that each attacker using its optimal disguised individual attack will maximize the magnitude of the disguised attack if the group of attackers act jointly. The following example with two attackers and $\ell = 2$ illustrates the issue.



(a) Optimal individual attacks.



(b) Optimal joint attack.

The optimal attack for v_1 is to point to u for v_2 is to point to w (red dotted arrows in (a)). However, if both attackers attack, then they should both point to u. This is generally true,

Theorem 6 *There is an optimal joint attack in which all attackers point to the same intermediate node u which is distance $\ell - 1$ from v_0.*

A detailed comparison of the optimal joint attack with the greedy strategy in which the attackers each adopt their individually optimal attacks is beyond the scope of this current paper.

4 Experimental results

In this section, we give some preliminary experimental results that quantify the effectiveness of Google bombs in various environments. There are four main degrees of freedom we explore: the nature of the graph, including its connectivity or edge density; the prominence (pagerank) of the attackers; the prominence of the victim; and, the value of α.

We ran our experiments on three types of graphs: *Random* is an Erdös-Reyni type ($G(n,p)$) random graph with edge probability p; BA (Barabási-Albert) is a preferential-attachment random graph with 5 outgoing edges per vertex [2]; (Such graphs are known to have power-law in-degree distributions, and since we add the vertices sequentially, there are no cycles.) $MWDTA$ is a modified "Winner's don't take all" random graph in which every node has at

least one out-going edge method [13]. (Such graphs are known to model certain characteristics of the world wide web graph such as power-law in and out-degree distributions.). The main difference between $MWDTA$ and BA random graphs is that in $MWDTA$, a larger number of nodes will have significant indegree, whereas in BA a few nodes have very large in-degrees. In order to make fair comparisons, we normalize graphs from different random graph models (*Random*, BA or $MWDTA$) to have the same expected number of edges.

First, we generate a random graph with 1,000 nodes, and randomly select 10 attackers and a victim. We then remove outgoing edges from the attackers and perform a pagerank computation, obtaining:

p_0, the page rank of the victim;
p_A, the average pagerank of the attackers;
$f_p(p)$, the pagerank distribution in the graph;
σ_p, the std. dev. of the pagerank distribution.

We only show results for two of the attacks described in Section 3.1: the optimal direct individual attack I, and the cycle attack C (the results for other suboptimal attacks are similar). Each attack is repeated a number of times on randomly generated graphs to increase the statistical significance of the results. We use the following measures of success for attack X,

$G(X) = \text{Gain} = \Delta p_0^X / p_0$,
$\bar{G}(X) = \text{Normalized Gain} = \Delta p_0^X / \sigma_p$,
$D(X) = \text{Discrepancy Factor} = G(I)/G(X)$.
$\bar{D}(X) = \text{Normalized Discrepancy} = \bar{G}(I) - \bar{G}(X)$.

The pagerank distribution $f_p(p)$ generally affects the effectiveness of an attack. Figure 1(a) shows pagerank distributions for the various random graphs. As can be seen, *Random* has a (near) Normal distribution, compared with BA and $MWDTA$ which have power-law type distributions in which $MWDTA$ appears to have a slightly fatter tail than BA.

Some detailed results on the effectiveness of the attacks are shown in Figure 1: (b) shows how connectivity (number of edges) in *Random* graphs with different p affects the attack; (c) shows different graph types; (b,c) show the dependence on the prominence of the attackers, and (d) on the prominence of the victim; (e) shows the dependence on α; and, (f) shows some results for the *rank* (as opposed to the pagerank). We give a summary of the results below.

Higher Density: All attacks decrease in magnitude (new edges have little additional effect when the graph is already dense).

Graph type: Prominence of attackers has (by far) the largest impact in *Random* graphs, then BA and $MWDTA$. (Pageranks in *Random* graphs are "concentrated" around the mean, so any bias in the victim's pagerank results in it becoming extreme. This is less so for BA and even less so for $MWDTA$.).

Higher Prominence of Attackers: Stronger attack.

Higher Prominence of Victim: Attacks become less effective and $D(C)$ decreases (diminishing returns).

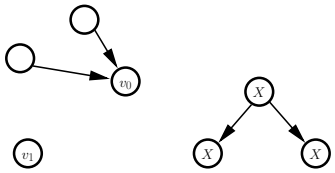
Lower α: $D(C)$ increases (it is more costly to divert from the individual attack).

Rank: For random graphs, an attack usually results in a top ranking for the victim, which is not usually the case for BA and $MWDTA$ graphs.

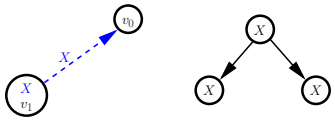
5 Discussion

We have shown that the best attack is the direct individual attack, in particular: *any* organized structure among the attackers reduces the impact of the attack; links that cycle back to attackers in an attempt to boost their pageranks are detrimental. The discrepancy between the optimal individual attack and suboptimal attacks can strongly depend on the graph type through the initial pagerank distribution. Our results indicate conditions that offer resistance to rank manipulation: dense, power-low type graphs in which victims already have high rank, attackers have low rank and α is small. Our analysis has been focused on increasing a page's rank (pagerank manipulation) in the entire graph, i.e., the victims rank is increased for *every* query. The underlying model is that the query identifies a set of nodes (based on text and anchor text), which defines an induced subgraph of the original graph. However, the nodes are ranked according to pagerank in the original graph. This model has the feature that pageranks do not need

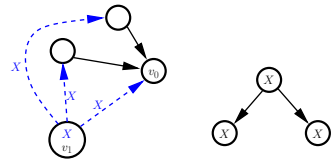
to be recomputed for the specific query. An alternative approach is to order the nodes with respect to the pageranks in the induced subgraph (hence these pageranks would need to be recomputed for every query). Such a model would mean that one attempts to boost the pagerank with respect to a specific query and not others. Our analysis does not apply to this model, and it is no longer true that the optimal attack is the direct individual attack. The following example (with a single attacker) illustrates the issue.



(a) Original graph.



(b) Direct attack.



(c) Indirect attack.

In (a) we show the original graph, where X will be the query text and the attacker wants to boost the rank of v_0 with respect to X. In (b) we show the subgraph induced by the direct attack, where the attacker places X in its page as well as in the anchor text of the link. In the resulting induced subgraph, the rank of v_0 is not the highest. The benefit of the non-direct attack in (c) is that other nodes that point to v_0 get included into the induced subgraph. Thus while the flow of rank from v_1 to v_0 is decreased, this is more than compensated for by the additional rank contribution from the newly included nodes. A better attack would arise if v_1 added another link to v_0. In fact for any attack in which v_1 has k links to v_1, a strictly better attack with $k+1$ links is possible. In this example, there is no optimal attack. In general, we can formulate this notion by saying that the attacker should add the minimum number of links to all nodes with paths to the victim which do not contain the query text, and hence would not be included in the subgraph. The attackers should then place as many parallel direct links as feasible. The end effect is to include all nodes with paths to the victim with a minimum diversion of page rank. Of course, such a huge attack is not very practical, and an interesting question is to consider the optimal attack under this model when each attacker has a fixed budget of links.

The PageRank algorithm favors attacks from groups that are not well connected, which makes it harder to detect the attack, and accountability in such an attack formation becomes an issue: who is responsible for the attack? Different variations of the PageRank algorithm may suffer a similar fate if they propagate the pagerank in a similar way (for example Topic-Sensitive PageRank [8], provided that the attacking group is considered relevant to the query). In order to avoid such a fate (a dilemma faced by any ranking method open to manipulation by small groups), either one must change the ranking function or somehow exclude the attacking group from the search engine's database. While such an approach is a reasonable way to deal with private companies attempting to manipulate rankings based on their own views, it is not very democracy-friendly to arbitrarily remove certain pages from a search engine.

As discussed in [6], the PageRank algorithm makes certain assumptions about the user navigation patterns and the web structure that may not apply to the Web anymore. [6] considers the effect of dangling nodes in the pagerank computation and provides methods to adjust for them. They also point out that users will rarely (if ever) navigate to one of several billion pages uniformly – they may not even know that these pages exist. In fact, users generally start from known sites and navigate from there. Hence, random navigation is more likely to bring them to one of these "anchor" sites. The HostRank algorithm [6] uses this assumption to choose a set of

anchor sites, and they show that such an approach is more resistant to attacks. A related issue is that of navigation along links from a site. One is more likely to trust a link on a highly ranked page, and one is more likely to follow a link to a highly ranked page. For example, it might be *much* more probable to follow one of the links from a search engine or a news Web site than a regular web page. The probability to navigate from a page in the PageRank algorithm is independent of a page's rank, and the link one selects to navigate is random. A plausible alternative is that the probability to navigate from a page should be proportional to the page's pagerank, and the probability to use a particular outgoing link is proportional to the pagerank of the destination page. Such a navigation model would lead to an equation (analogous to (1)) of the form

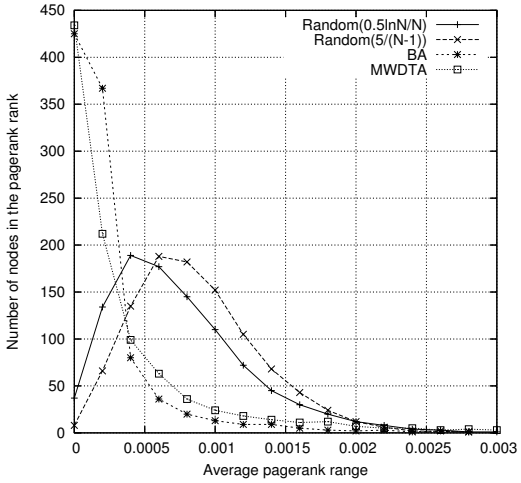
$$p_i = \kappa\alpha p_i \sum_{(v_j,v_i)\in E} \frac{p_j^2}{\sum_{(v_j,v_k)\in E} p_k} + \frac{1-\alpha}{N}.$$

More effort could be spent how the transition probabilities generally affect the pageranks and their manipulability. [6] discusses such issues for nodes with unknown outgoing links and [15] uses the amount of traffic flow through the nodes to model the transition probabilities. It would be interesting to see what the optimal attack with such ranking algorithms is. In short, objective methods for the selection of the anchor sites or more plausible navigation models deserves closer examination. One must also bear in mind (see for example [6]) that the computational complexity of the algorithm is also an important practical consideration for any ranking algorithm.

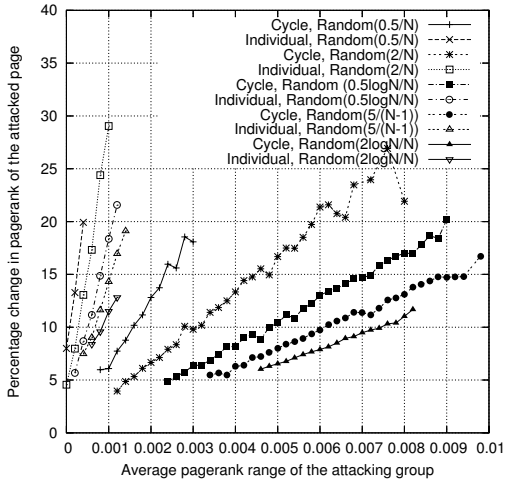
Other factors, which we do not study here, might be significant to the success of an attack. [7] argues that anchor text pointing to a page gives information regarding the subject matter of that page, and relationships between different pages. For example, Google may consider both the pagerank and the frequency of keywords in links pointing to a page when computing the score of the page. Google bombs in the past used the same keywords when pointing to the attacked page, i.e., the bombing links were correlated in that they all had the same keywords, whereas in general, links pointing to a website would not display such a correlation. If some linear combination of these two factors is then used in the final score, it will favor attacks over the natural Web behavior. If some small group of sites use a specific keyword to point to a victim, it is unlikely that this groups's sites are unrelated, and one could (for example) add pseudo-links among these sites, since the expection would be that they participate in some group structure. As our results show, these pseudo-links will reduce the magnitude of the attack. One could go so far as to say that if after the addition of such pseudo-links in the graph, the pagerank distribution does not change significantly, then the ranking algorithm should be more resistant to manipulation.

The analysis of the optimal attack structure provides a new tool for looking at resistance to link manipulation. Such metrics and an understanding of optimal attack formations for other algorithms should be fruitful directions for future work.

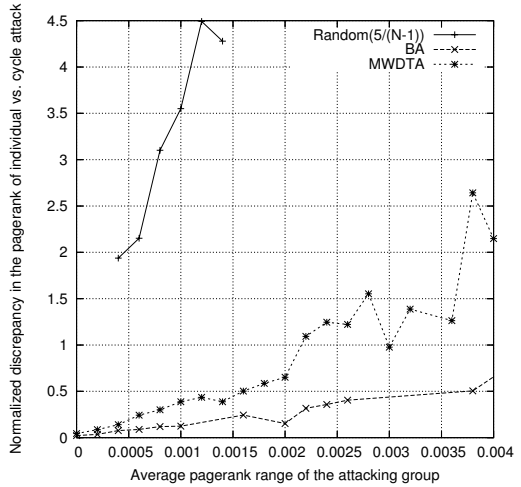
Acknowledgement. We are grateful to Mark Goldberg for his initial feedback on this work.



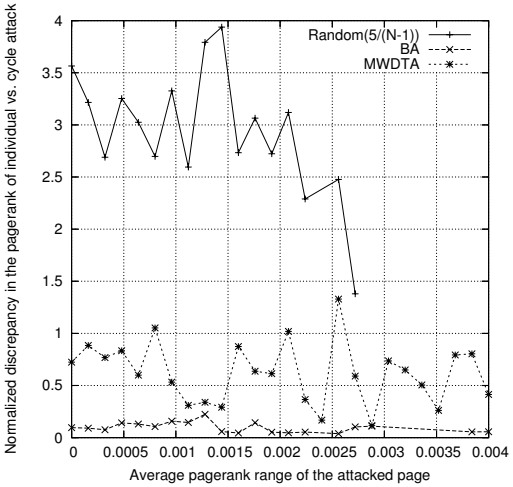
(a) Pagerank distributions of different graphs



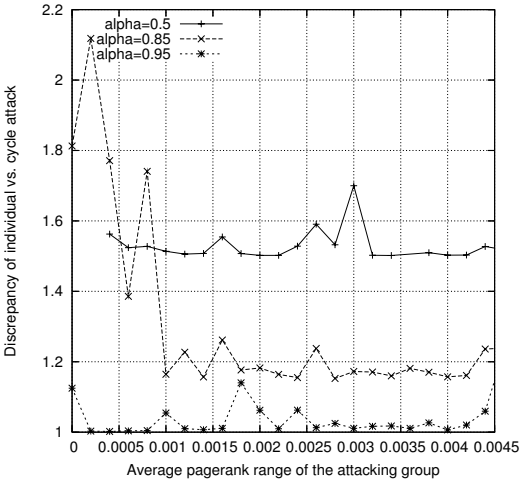
(b) Graphs with different edge densities.



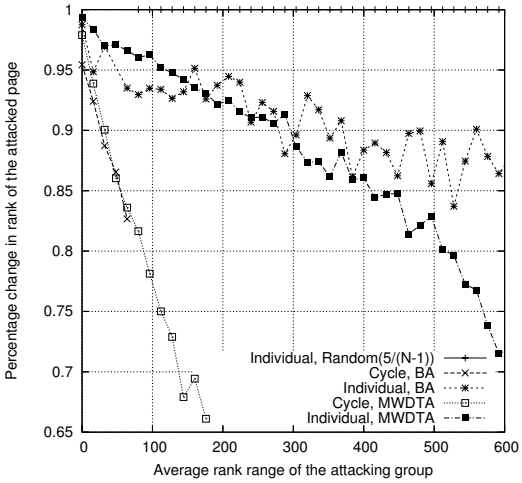
(c) Attacks in different graph types



(d) Dependence on pagerank of the victim



(e) Dependence on alpha (α) for MWDTA graphs



(f) Change in rank of victim

Figure 1: Experimental Results for $n = 1000$.

References

[1] S. Adali, T. Liu and M. Magdon-Ismail, *An analysis of optimal link bombs*, CS Department Technical Report, TR#05–11, RPI, Troy, NY, 2005.

[2] A.-L. Barabàsi and R. Albert. *The Emergence of Scaling in Random Networks*, Science, 286, 1999

[3] S. Brin and L. Page. *The Anatomy of a Large-Scale Hypertextual Web Search Engine*, Proc. 7th WWW Conf., 1998

[4] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. *Graph structure in the web*, Computer Networks, 33(1-6): pp. 309–320, 2000

[5] S. Chien, C. Dwork, R. Kumar, D. Simon, and D. Sivakumar, *Towards exploiting link evolution*, Workshop on Algorithms for the Web, 2002.

[6] N. Eiron, K. S. McCurley, J. A. Tomlin, *Ranking the Web Frontier*, Proc. 13th WWW Conf., 2004.

[7] N. Eiron and K. S. McCurley. *Analysis of anchor text for web search*, Proc. 26th SIGIR, 2003.

[8] T. H. Haveliwala. *Topic-sensitive pagerank*, Proc. 11th WWW Conf., 2002.

[9] http://www.microcontentnews.com/articles/googlebombs.htm

[10] T. McNichol. *Engineering Google Results to Make a Point*, NY Times, Jan 22, 2004.

[11] A. Y. Ng, A. X. Zheng, and M. I. Jordan, *Stable algorithms for link analysis*, Proc. 24th SIGIR 2001.

[12] L. Page, S. Brin, R. Motwani, and T. Winograd. *The pagerank citation ranking: Bringing order to the web*, Stanford University Database Group TR, 1998.

[13] D. M. Pennock, G. W. Flake, S. Lawrence, E. J. Glover, and C. L. Giles. *Winner's Don't Take All*, Proc. National Academy of Sciences, 99(8), 2002.

[14] D. Sullivan. *Google's (and Inktomi's) Miserable Failure*, searchenginewatch.com, Jan 6, 2004.

[15] J. Tomlin, *A New Paradigm for Ranking Pages on the World Wide Web*, Proc. 12th WWW Conf., 2003.

[16] H. Zhang, A. Goel, R. Govindan, K. Mason, and B. Van Roy, *Making eigenvector-based reputation systems robust to collusion*, Workshop on Algorithms and Models for the Web Graph (WAW), 2004.

Web Spam, Propaganda and Trust

Panagiotis T. Metaxas
Computer Science Department
Wellesley College
Wellesley, MA 02481, USA
pmetaxas@wellesley.edu

Joseph DeStefano
Math and Computer Science Department
College of the Holy Cross
Worcester, MA 01610, USA
joed@mathcs.holycross.edu

ABSTRACT

Web spamming, the practice of introducing artificial text and links into web pages to affect the results of searches, has been recognized as a major problem for search engines. It is also a serious problem for users because they are not aware of it and they tend to confuse trusting the search engine with trusting the results of a search.

In this paper, we first analyze the influence that web spam has on the evolution of the search engines and we identify the strong relationship of spamming methods to propagandistic techniques in society. Our analysis provides a foundation to understanding why spamming works and offers new insight on how to address it. In particular, it suggest that one could use anti-propagandistic techniques in the web to recognize spam. The second part of the paper demonstrates such a technique, called backwards propagation of distrust.

In society, recognition of an untrustworthy message (in the opinion of a particular person or other social entity) is a reason for questioning the entities that recommend the message. Entities that are found to strongly support untrustworthy messages become untrustworthy themselves. So, social distrust is propagated backwards for a number of steps. Our algorithm simulates this social behavior on the web graph.

In our algorithm, starting from an untrustworthy (according to the end user) site s, we examine its *trust neighborhood*, that is, the neighborhood of sites that link to s in a few steps. Evaluating the sites-members of the neighborhood we identify a biconnected component (BCCs) with a high percentage of untrustworthy sites. BCCs are formed when there are multiple paths to reach s, thus indicating a concerted effort to promote s. This is not the case when starting from a trustworthy site.

Our tool explores thousands of nodes within minutes and could be deployed at the browser-level, making it possible to resolve the moral question of who should be making the decision of weeding out spammers in favor of the end user.

Our approach can lead to browser-level web spam filters that work in synergy with the powerful search engines to deliver personalized, trusted web results.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.3.m [**Information Storage and Retrieval**]: Miscellaneous

AIRWeb2005, May 10, 2005, Chiba, Japan.

General Terms

Algorithms, Experimentation, Social Networks, Propaganda, Trust

Keywords

search, Web graph, link structure, PageRank, HITS, Web spam

1. INTRODUCTION

Web spamming is often defined as the practice of manipulating web pages in order to cause search engines to rank some web pages higher than they would without any manipulation[1]. Spammers aim at search engines, but target the end users. Their motive is usually commercial, but can also be political, or religious.

One of the reasons behind the users' difficulty to distinguish trustworthy from untrustworthy information comes from the success that both search engines and spammers have enjoyed in the last decade. Users have come to trust search engines as a means of finding information, and spammers have successfully managed to get them to transfer that trust to the results of each search.

From their side, the search engines have put considerable effort in delivering spam-free query results and have developed sophisticated ranking strategies. Two such ranking strategies that have received major attention are the well-known PageRank [6] and HITS [28] algorithms. Achieving high PageRank has become a sort of obsession for many companies' IT departments, and the *raison d'être* of spamming companies. Some estimates indicate that at least 8% of all pages indexed is spam [12] while experts consider web spamming the single most difficult challenge web searching is facing today. [23]. Search engines typically see web spam as an interference to their operations and would like to restrict it, but there can be no algorithm that can recognize spamming sites based solely on graph isomorphism [5].

First, however, we need to understand why spamming works beyond the technical details, because spamming is a social problem first, then a technical one. In this paper we show its extensive relationship to social propaganda, and evidence of its influence on the evolution of search engines.

[1]We should mention here that there is not a complete agreement on the definition of web spam among authors, which leads to some confusion. Moreover, to people unfamiliar with web spam, the term is mistaken for email spam. A more descriptive name for it would be "search engine ranking manipulation."

Our approach can explain the reasons web spamming has been so successful and suggest new algorithmic ways of dealing with it. Finally, we discuss what we believe should be a frame for the long-term approach to web spam.

The rest of this paper is organized as follows. The next section gives an overview of the problem of web spamming and information reliability for a general audience. Section 3 discussed the relationship between webgraph and the trust social network while the following section analyzes the evolution of search engines as their response to spam. Section 5 describes the backward propagation of distrust method and the following section presents some of our experimental results running this algorithm. Section 7 discusses some related research and the final section has our conclusions and some discussion of future directions of this work.

2. BACKGROUND

The web has changed the way we inform and get informed. Every organization has a web site and people are increasingly comfortable accessing it for information for any question they may have. The exploding size of the web necessitated the development of search engines and web directories. Most people with online access use a search engine to get informed and make decisions that may have medical, financial, cultural, political, security or other important implications [10, 40, 24, 32]. Moreover, 85% of the time, people do not look past the first ten results returned by the search engine [38]. Given this, it is not surprising that anyone with a web presence struggles for a place in the top ten positions of relevant web search results. The importance of the top-10 placement has given birth to a new industry, which claims to sell know-how for prominent placement in search results and includes companies, publications, and even conferences. Some of them are willing to bend the truth in order to fool the search engines and their customers, by creating web pages containing web spam [12].

The creators of web spam are often specialized companies selling their expertise as a service, but can also be the web masters of the companies and organizations that would be their customers. Spammers attack search engines through text and link manipulations [23, 19]:

- **Text spam**: This includes excessively repeating text and/or adding irrelevant text on the page that will cause incorrect calculation of page relevance; adding misleading meta-keywords or irrelevant "anchor text" that will cause incorrect application of rank heuristics.
- **Link spam**: This technique aims to change the perceived structure of the webgraph in order to cause incorrect calculation of page reputation. Such examples are the so-called "link-farms", "mutual admiration societies", page "awards", domain flooding (plethora of domains that re-direct to a target site), etc.

Both kinds of spam aim to boost the ranking of spammed web pages. Sometimes **cloaking** is included as a third spamming technique [23, 20]. Cloaking aims to serve different pages to search engine robots and to web browsers (users). These pages could be created statically or dynamically. Static pages, for example, may employ hidden links and/or hidden text with colors or small font sizes noticeable by a crawler but not by a human. Dynamic pages might change content on the fly depending on the visitor, submit millions of pages to "add-URL" forms of search engines, etc. We consider the false links and text themselves to be the spam, while, strictly speaking, cloaking is not spam, but a tool that helps spammers hide their attacks.

Since anyone can be an author on the web, these practices have naturally created a question of *information reliability*. An audience used to trusting the written word of newspapers and books is unable, unprepared or unwilling to think critically about the information obtained from the web. A recent study [17] found that while college students regard the web as a primary source of information, many do not check more than a single source, and have trouble recognizing trustworthy sources online. In particular, two out of three students are consistently unable to differentiate between facts and advertising claims, even "infomercials." Very few of them would double-check for validity. At the same time, they have considerable confidence in their abilities to distinguish trustworthy sites from non-trustworthy ones, especially when they feel technically competent. We have no reason to believe that the general public will perform any better than well-educated students. In fact, a recent analysis of internet related fraud by a major Wall Street law firm [10] puts the blame squarely on the investors for the success of stock fraud cases.

3. THE WEBGRAPH AS A SOCIAL NET

The web is typically represented by a directed graph [8]. The nodes in the webgraph are the pages (or sites) that reside on servers on the internet. Arcs correspond to hyperlinks that appear on web pages (or sites). *Web spammers are trying to alter the web graph in ways beneficial to them.*

The theory of social networks of Trust [41] also uses directed graphs to represent relationships between social entities. The nodes correspond to social entities (people, institutions, ideas). Arcs correspond to recommendations between the entities they connect. *Propagandists are trying to alter the trust social net in ways beneficial to them.*

This connection is more than just a similarity in descriptions. The web itself is a social creation, and both PageRank and HITS are socially inspired ranking algorithms. [6, 28, 36]. Socially inspired systems are subject to socially inspired attacks, however. Not surprisingly then, the theory of propaganda detection [31] can provide intuition into the dynamics of the web graph. First developed in the beginning of World War II by the Institute for Propaganda Analysis [15, 31], the theory of propaganda detection identifies several techniques that propagandists often employ in order to manipulate perception. Name calling, glittering generalities, testimonial, bandwagon and transfer are the more well-known of them.

PageRank is based on the assumption that the reputation of an entity (a web page in this case) can be measured as a function of both the number and reputation of other entities linking to it. A link to a web page is counted as a "vote of confidence" to this web site, and in turn, the reputation of a page is divided among those it is recommending [2]. The implicit assumption is that hyperlink "voting" is taking place independently, without prior agreement or central

[2]Since HTML does not provide for "positive" and "negative" links, all links are taken as positive. This is not always true, but is considered a reasonable assumption. Recently, Google introduced the "nofollow" attribute for hyperlinks, but it is very unlikely that web spammers will use it.

Graph Theory	*Web Graph*	*Trust Social Network*
node	web page or site	social entity
node weight	rank (accord. to SE)	reputation (accord. to user)
node weight computation	ranking formula	based on top recommenders
	automatic	on demand
arc	hyperlink	trust opinion
arc meaning	vote of confidence	recommendation
arc weight	degree of confidence	degree of entrustment
arc weight computation	ranking formula	arbitrary, semi-consistent
arc weight range	$[0 \dots 1]$	$[distrust \dots trust]$

Table 1: Graph theoretic correspondence between the Webgraph and the Trust Social Network.

control. Spammers, like social propagandists, form structures that are able to gather a large number of such "votes of confidence" by design, thus breaking the assumption of independence in a hyperlink.

Table 1 has the correspondence between graph theoretic terms, the web graph according to a search engine, and the trust social network of a particular user.

4. EVOLUTION OF SEARCH ENGINES

In the early 90's, when the web numbered just a few million servers, the **first generation** search engines were ranking search results using classic information retrieval techniques: the more rare words two documents share, the more similar they are considered to be. [37, 22] A search query Q is simply a short document and the results of a search for Q are ranked according to their (normalized) similarity to the query.

The first attack to this "$tf.idf$ ranking," as it is known, came from within the search engines. Around 1995, search engines started selling search keywords to advertisers as a way of generating revenue: If a search query contained a "sold" keyword, the results would include targeted advertisement and a higher ranking for the link to the sponsor's web site. This is the first time we have a socially inspired ranking, which follows marketing practices of the real world.

Mixing search results with paid advertisement raised serious ethical questions, but also showed the way to financial profits to spammers who started their own attacks by creating pages containing many rare keywords to obtain a higher ranking score. In terms of propaganda theory, the spammers employed a variation of the technique of *glittering generalities* to confuse the first generation search engines [31, pg. 47]:

The propagandist associates one or more suggestive words without evidence to alter the conceived value of a person or idea.

To avoid spammers search engines would keep secret their exact ranking algorithm. Secrecy is no defense, however, since secret rules were figured out by experimentation and reverse engineering. (e.g., [35, 33]).

Second generation search engines started employing more sophisticated ranking techniques in an effort to nullify the effects of glittering generalities. One of the more successful techniques was based on the "link voting principle": Each web site s has value equal to its "popularity", which is influenced by the set B_s of sites pointing to site s. Lycos became the champion of this ranking technique and had its own popularity skyrocket around 1996.[34]. Doing so, it was also distancing itself from the ethical questions introduced by combining advertising with ranking.

Unfortunately, this ranking method did not succeed in stopping spammers either. Spammers started creating clusters of interconnected web sites that had identical or similar contents with the site they were promoting, which subsequently became known as "link farms" (LF). The link voting principle was socially inspired, so spammers used the well known propagandistic method of *bandwagon* to circumvent it [31, pg. 105]:

With it, the propagandist attempts to convince us that all members of a group to which we belong are accepting his program and that we must therefore follow our crowd and "jump on the band wagon".

Similarly, the spammer is promoting the impression of a high degree of popularity by inter-linking many internally controlled sites that will eventually all share high ranking.

The introduction of PageRank in 1998 was a major development for search engines, because it seemed to provide a more sophisticated anti-spamming solution. Under PageRank, not every link contributes equally to the "reputation" of a page. Instead, links from highly reputable pages contribute much higher than links from other sites. That way, the site networks developed by spammers would not influence much their PageRank, and Google became the search engine of choice. HITS is another socially-inspired ranking which has also received a lot of attention. [28]. The HITS algorithm divides the sites related to a query between "hubs" and "authorities". Hubs are sites that contain many links to authorities, while authorities are sites pointed to by the hubs and they both gain reputation.

PageRank and HITS marked the development of the **third generation** search engines [3]. Unfortunately, spammers have again found ways of circumventing them. In PageRank, a page enjoys absolute reputation: its reputation is not restricted on some particular issue. Spammers deploy sites with expertise on irrelevant subjects, and they justifiably acquire high ranking on their expert sites. Then they bandwagon their networked sites with the expert sites, creating a "mutual admiration society" (MAS). This is the well-known propagandistic technique of *testimonials* [31, pg. 74]:

Well known people (entertainers, public figures, etc.) offer their opinion on issues about which they are not experts.

HITS has also shown to be highly spammable by this tech-

[3][7] considers the search engines in our 2nd and 3rd generation to be in the same group. We believe that both the ranking and attack methods put them in different categories.

nique due to the fact that its effectiveness depends on the accuracy of the initial neighborhood calculation.

The table below summarizes our findings for the first three generations of search engines and the correspondence between web spam and social propaganda.

SE	Ranking	Spamming	Propaganda
1st Gen	Doc Similarity	keyword stuffing	glittering generalities
2nd Gen	+ Site popularity	+ link farms	+ bandwagon
3rd Gen	+ Page reputation	+ mutual admiration societies	+ testimonials

Web search corporations are reportedly busy developing the engines of the next generation [7]. The new search engines hope to be able to recognize "the need behind the query" of the user. Given the success the spammers have enjoyed so far, one wonders how will they spam the fourth generation engines. Is it possible to create a ranking that is not spammable? Put another way, can the web as a social space be free of propaganda? Seen in this light, it appears that we are trying to create in cyberspace what societies have not succeeded in creating in their social space. This may not be possible. However, we can learn to live in a web with spam as we live in society with propaganda, given appropriate education and technology.

5. AN ANTI-PROPAGANDISTIC METHOD

Web Spam seems to be the driving force behind the evolution of search engines in their effort to provide quality results. So far, the battle with web spam is only waged at the search engine level, though the end users are the ones affected directly by it. When users query a popular search engine for questions that happen to be the target of unreliable advertisement (e.g., "Can human growth hormone increase muscle mass?") or happen to be controversial in nature (e.g., "is ADHD a real disease?"), they find plethora of responses that can be considered untrustworthy. For example, the first query provides almost exclusively links to human growth hormone (hGH) products that, among other benefits, would significantly increase muscle mass without increased exercise, decrease fat without change in diet or habits, enhance sexual performance, increase the good cholesterol while decreasing the bad, re-grow hair, decrease blood pressure, remove wrinkles, and increase memory retention. Similarly, in the second query one finds an unbalanced view of attention-deficit, hyperactivity disorder (ADHD) that does not include the opinion of major institutions such as the American Psychiatric Association or clinicians in major research universities. To the inexperienced user it may appear that the search engine promotes untrustworthy, unreliable or unbalanced views. What really happens, of course, is that these queries have been the target of spammers.

Since spammers employ propagandistic techniques, it makes sense to design anti-propagandistic methods for defending against them. These methods need to be user-initiated. We are considering trustworthiness to be a personal decision, not an absolute quality of a site. One person's gospel is another's political propaganda, and our goal is to design methods that help individuals make more informed decisions about the quality of the information they find on the web.

Here is one way that people defend against propaganda in every day life:

In society, when an untrustworthy recommendation is detected, it gives us a reason to reconsider the trustworthiness of the recommender. Recommenders who strongly support an untrustworthy recommendation become untrustworthy themselves.

This process is selectively repeated a few times, propagating the distrust backwards to those who strongly support the recommendation. The results of this process become part of our belief system and are used to filter future information.

We set out to test whether a similar process might work on the web. Our algorithm takes as input the URL of the server s containing a page that the user determined to be untrustworthy. This page could have come to the user through web search results (like the ones above) or via the suggestion of some trusted associate (e.g., a society that the user belongs to).

Starting from s we build a breadth-first search (bfs) tree of the sites that link to s in a few "clicks" (Figure 1). We do not explore the web neighborhood directly in this step. Instead, we use the Google API [16] for finding the backlinks. We call the directed graph that is revealed by the backlinks, the "trust neighborhood" of s.

The question arises on whether we should distrust all of the sites in the trust neighborhood of s or not. Is it reasonable to become suspicious of every site pointing to s in a few steps? They are "voting in confidence" after all. Such a radical approach is not what we do in everyday life. Rather, we selectively propagate distrust only to those that most strongly support an untrustworthy recommendation. Thus, we decided to take a conservative approach and examine only those sites that show a more concerted effort in supporting s. In particular, we focused on the biconnected component (BCC) that includes s (Figure 2).

A BCC is a graph that cannot be broken into disconnected pieces by deleting any single vertex. An important characteristic of the BCC is there are at least two independent paths from any of its vertices to s. Strictly speaking, the BCC is computed on the undirected graph. But since the trust neighborhood is generated through the bfs, the cross edges (in bfs terminology) create cycles in the undirected graph (Figure 1). Each cycle found in the BCC must have at least one "ring leader", from which there are two directed paths to s, one leaving through the discovery edge and the other through the cross edge. We view the existence of multiple paths from ring leaders to s as evidence of strong support of s. The BCC reveals the members of this support group.

More formally, the algorithm is as follows:

```
Input: Untrustworthy site s.

S = {s}
Using BFS for depth D do:
   Find the set U of sites linking to sites in S
      using the Google API (up to B backlinks / site)
   Ignore blogs, directories, edu's
   S = S + U

Compute and output the BCC of S that includes s
```

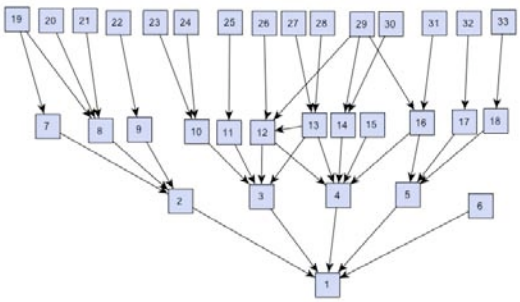


Figure 1: An example of a breadth-first search tree in the trust neighborhood of site 1. Note that some nodes (12, 13, 16 and 29) have multiple paths to site 1. We call these nodes "ring leaders" that show a concerted effort to support 1.

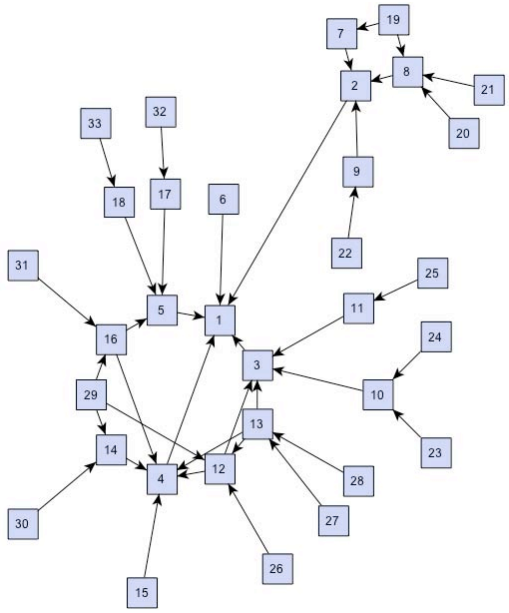


Figure 2: The BCC of the trust neighborhood of site 1 is drawn in a circular fashion for clarity.

To be able to implement the above algorithm at the browser side, we restrict the following parameters: First, the BFS's depth D is set to 3. We are not interested in exploring a large chunk of the web, just a small neighborhood around s. Second, we limit the number B of backlink requests from the Google API to 30 per site. Finally, we introduced in advance a set of *stop sites* that are not to be explored further. A stop site is one that should not be included in the trust neighborhood either because the trustworthiness of such a site is irrelevant, or because it cannot be defined. In the first category we placed URLs of educational institutions (domains ending in .edu). Academicians are not in the business of pointing to commercial sites. When they do, they do not often convey trust in the site. In the latter we placed a few well known Directories (URLs ending in yahoo.com, dmoz.org, etc.) and Blog sites (URLs containing the string 'blog' or 'forum'). Anyone can put an entry into an unsupervised blog or directory. No effort to create an exhaustive list of blogs or directories was made.

With these restrictions, our algorithm can be implemented on an average workstation and produce graphs with up to a few thousand nodes within minutes. Note that the slowest step is the query of the backlinks. More recently, a threaded version of the program can explore several thousand sites in minutes.

6. EXPERIMENTAL RESULTS

In our experiments, we examined the trust graphs of eight untrustworthy and two trustworthy sites, collected from the search results of the first hGH query and of query "Benefits of Calcium supplements". In the table 2 below these sites are labeled as U-1 to U-8 and T-1 to T-2, respectively. See Figure 3 for an example of one such site (U-1). We run the experiments between September 17 and November 5, 2004. We should note here that all sites have comparable PageRank. In fact, all but U-1 and T-1 have PageRank 5. The remaining two sites have PageRank 6. (Pageranks were recorded at the time of the experiments.)

To determine the trustworthiness of each site we had an evaluator look at a sample of the sites of the BCC. Due to the significant manual labor involved, only 20% of the total 1,396 BCC sites were sampled and evaluated. To select the sample sites, we employed stratified sampling with skip interval 5. The stratum used was similarity of the site to the starting site.

Each site in the sample was classified as either Trustworthy, Untrustworthy, or Non-determined. The last category includes a variety of sites for which the evaluator could not clearly classify due to the language used in the site, the subject matter, or the fact that a Blog or Directory can not fall simply into one of the U/T categories. (Not every blog contains the string "blog" in their URL.)

The experiments show that the trustworthiness of the starting site was a very good predictor for the trustworthiness of the BCC sites. In fact, there were very few trustworthy sites in the trust graph of sites U-1 to U-8. As one might expect, a trustworthy site is unlikely to deliberately link to an untrustworthy site, or even to a site that "associates" itself with an untrustworthy one. In other words, the "vote of confidence" analogy holds true for sites that are responsibly

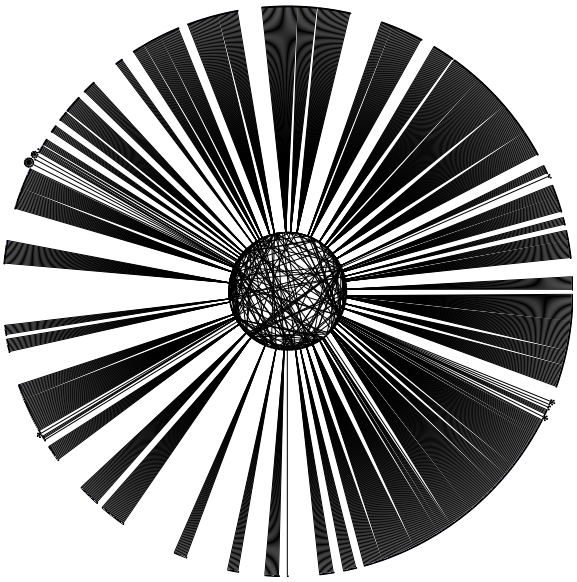


Figure 3: The trust graph of starting site U-1. The circularly drawn nodes in the middle form its largest biconnected component. This experiment found a trust graph of 1307 sites, 228 of which were connected with 465 edges into a BCC. Only 2% trustworthy sites were found in the BCC, while 74% of them were untrustworthy. The remaining sites were mostly directories (13%) or other non-determined sites.

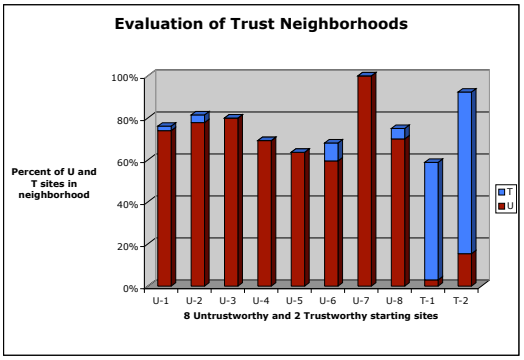


Figure 4: Histogram of the results in table 2.

choosing their links. On the other hand, the analogy is not as strong when starting from a trustworthy site, since untrustworthy sites are free to link to whomever they choose. After all, there is some value in portraying a site in good company. Users may be tempted to conclude that, if a site points to "good" sites, it must be "good" itself – another well-known propagandistic technique. Yet, spammers are unlikely to link to too many sites outside their spamming network in order to avoid "leaking" PageRank [5].

Research in the past has focused on the identification of web communities through the use of bipartite cores [29] or maximum flow in dense subgraphs [14]. These ideas do not apply to our construction. For one, we are not trying to identify a community of the starting site, but a sample of its trust neighborhood. In fact, we never look at the links coming out of s (or any other site) directly. One of the benefits of our method is that we do not need to explore the web graph explicitly, which would be impossible for a client computer.

7. RELATED WORK

Web spamming has received a lot of attention lately [1, 3, 4, 5, 12, 13, 20, 22, 23, 25, 29, 32, 33, 35]. The first papers to raise the issue were [33, 23]. The spammers' success was noted in [4, 10, 12, 13, 17, 24]. Web search was explained in [2]. The related topic of cognitive hacking was introduced in [11].

Characteristics of spamming sites based on diversion from power laws are presented in [12]. Current tricks employed by spammers are detailed in [19]. An analysis of the popular PageRank method employed by many search engines today and ways to maximize it in a spamming network is described in [5]. TrustRank, a modification to the PageRank to take into account the evaluations of a few seed pages by human editors, employees of a search engine, is presented in [20]. Techniques for identifying automatically link farms of spam pages will be presented in [42].

A comprehensive treatment on social networks is presented in [41]. The connection between the Web and social networks was explicitly noted in [30, 36] and implicitly used

S	$\|V_G\|$	$\|E_G\|$	$\|V_{BCC}\|$	$\|E_{BCC}\|$	**Trust.**	**Untr.**
U-1	1307	1544	228	465	2%	74%
U-2	1380	1716	266	593	4%	78%
U-3	875	985	97	189	0%	80%
U-4	457	509	63	115	0%	69%
U-5	716	807	105	189	0%	64%
U-6	312	850	228	763	9%	60%
U-7	81	191	32	143	0%	100%
U-8	1547	1849	200	430	5%	70%
T-1	1429	1566	164	273	56%	3%
T-2	241	247	13	17	77%	15%

Table 2: Sizes of the explored graphs and their BCC's for eight untrustworthy (U-1 to U-8) and two trustworthy (T-1 and T-2) starting sites. Column $|V_G|$ contains the number of vertices that our algorithm found in the trust neighborhood of starting site s (starting from site s and exploring in breadth-first search the backlinks of s. Column $|E_G|$ has the number of edges in the trust neighborhood. Columns $|V_{BCC}|$ and $|E_{BCC}|$ contains the numbers of edges of the largest biconnected component within G. The last two columns contain the estimated percentages of trustworthy and untrustworthy sites found in the BCCs. 20% of each BCC were evaluated using stratified sampling with stratum a site's similarity to the starting site.

in [6, 28]. In fact, Kleinberg's work explores many of these connections (e.g., [27]). Identification of web communities was explored in [29, 14]. Propagation methods for trust and distrust are discussed in [18]. Work on topic-sensitive and personalized web search is presented in [21, 26]. The effect that search engines have on page popularity was discussed in [9].

8. CONCLUSIONS

In this paper we have argued that web spam is to cyberworld what propaganda is to society. As far as we know, this is the first time this relationship is noted. As evidence of the importance of this analogy, we have shown that the evolution of search engines can be simply understood as the search engines' response defending against spam.[4] New search engines are not invented every few years, as it is sometimes reported; they are developed when researchers have a good answer to spam.

Further, our findings suggests that anti-spamming techniques can now be developed by mimicking anti-propagandistic methods. In particular, we have presented automatic ways of recognizing trust graphs on the web based on the biconnected component around some starting site. Experimental results from a number of such instances show our algorithm's ability of recognizing parts of a spamming network.

With such results, the question arises as to what one should do once one recognizes a spamming network. This is a question that has not attracted much attention in the past. The default approach is that a search engine would delete such networks from its indices [12] or might downgrade them by some prespecified amount [20].

Both of these approaches, however, require a universal agreement of what constitutes spam. Such an agreement cannot exist; one person's spam may be another person's treasure. Should the search engines determine what is trustworthy and what is not? Willing or not, they are the *de facto* arbiters of what information users see [39]. As in a popular cartoon, a kid responds to the old man who has been looking all his life for the meaning of life: "If it is not on Google or eBay, it does not exist."

We believe that it is the users' right and responsibility to decide what is acceptable for them. Their browser, their window to cyberworld, should enhance their ability to make this decision. User education is fundamental: People should know how search engines work and why, and how information appears on the web. But they should also have a browser that can help them determine the validity and trustworthiness of information.

The tool we described in an earlier section is a first step in this direction. Ultimately, it would be used along with a set of trust certificates that contains the portable trust preferences of the user, a set of preferences that the user can accumulate over time. Organizations that the user joins and trusts may also add to this set. A combination of search engines capable of providing indexed content and structure [21], including identified neighborhoods, with a browser capable of filtering those neighborhoods through the user's trust preferences, would provide a new level of reliability to the user's information gathering. Sharing ranking decisions with the end user will make it much harder for spammers to tune to a single metric.

8.1 Future Work

In our experiments we also devised a simple method to evaluate the similarity of the contents of each site to the starting site s. After the trust neighborhood was explored, we fetched and concatenated a few pages from each site (randomly choosing from the links that appeared in the domain URL) into a document. Then, we tried to determine the similarity of each such document to the document of the starting site. Similarity was determined using the $tf.idf$ ranking on the universe of the sites explored. We are aware that having a limited universe of documents does not give the best similarity results, but we wanted to get a feeling of whether our method could further be used to distinguish between "link farms" and "mutual admiration societies". The initial results were encouraging (see Fig. 5), showing a higher percentage of untrustworthy sites among those most similar to s. Nevertheless, more work is needed in this area.

[4]We do not imply here that web spam is the *sole* force behind the evolution of the search engines, but that it is a dominant one.

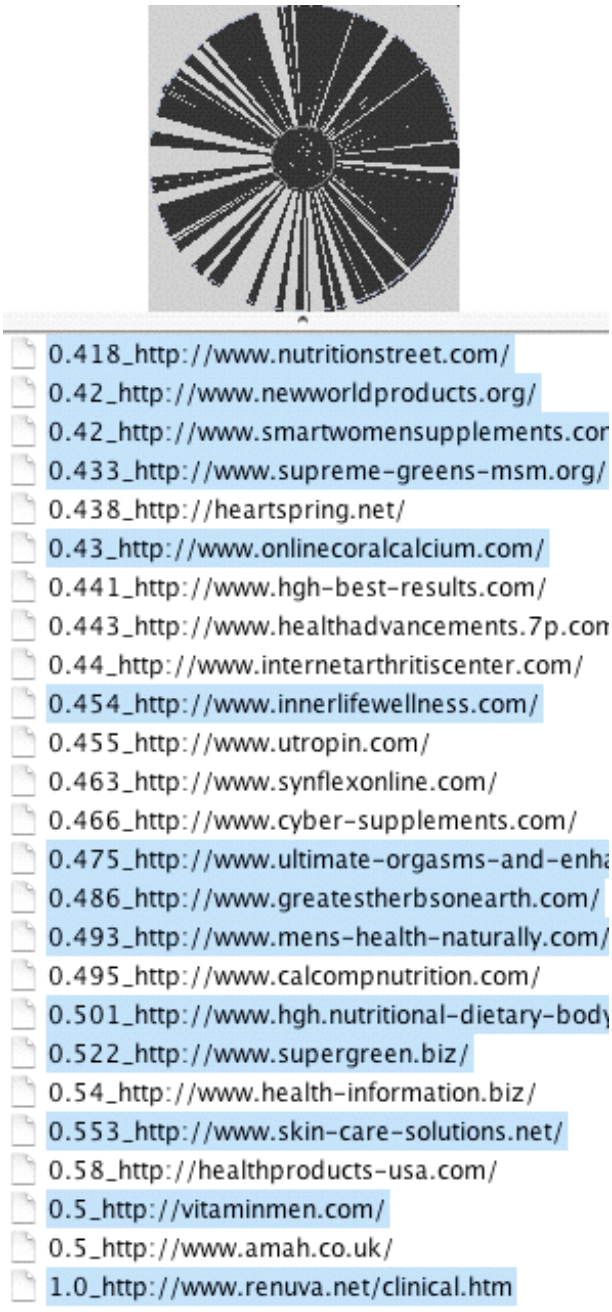


Figure 5: The list of sites similar to the starting site U-1 (at the end of the list). The hilited sites are those that participate in the BCC. The number in front of the URL corresponds to its calculated similarity to the starting site.

Several possible extensions can be considered in this work. Generating graphs with more backlinks per site, comparing the evolution of trust neighborhoods over time, examining the density of the BCCs, and finding a more reliable way to compute similarity are some of them. We also expect that the results would be strengthened if one considers the triconnected (or higher) components of the trust neighborhood.

9. ACKNOWLEDGEMENTS

The authors would like to thank Mirena Chausheva, Meredith Beaton-Lacoste, Scott Anderson and Scott Dynes for their valuable contributions. They would also like to thank David "Pablo" Cohn and the anonymous referees for their suggestions. The graphs shown in this paper were drawn using the yEd package [43].

10. REFERENCES

[1] B. Amento, L. Terveen, and W. Hill. Does authority mean quality? Predicting expert quality ratings of web documents. In *Proceedings of the Twenty-Third Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2000.

[2] A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, and S. Raghavan. Searching the web. *ACM Transactions on Internet Technology*, 1(1):2–43, June 2001.

[3] K. Bharat, A. Z. Broder, J. Dean, and M. R. Henzinger. A comparison of techniques to find mirrored hosts on the WWW. *Journal of the American Society of Information Science*, 51(12):1114–1122, 2000.

[4] K. Bharat, B.-W. Chang, M. R. Henzinger, and M. Ruhl. Who links to whom: Mining linkage between web sites. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 51–58. IEEE Computer Society, 2001.

[5] M. Bianchini, M. Gori, and F. Scarselli. PageRank and web communities. In *Web Intelligence Conference 2003*, Oct. 2003.

[6] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[7] A. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002.

[8] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. *Comput. Networks*, 33(1-6):309–320, 2000.

[9] J. Cho and S. Roy. Impact of search engines on page popularity. In *WWW 2004*, May 2004.

[10] T. S. Corey. Catching on-line traders in a web of lies: The perils of internet stock fraud. Ford Marrin Esposito, Witmeyer & Glesser, LLP, May 2001. http://www.fmew.com/archive/lies/.

[11] G. Cybenko, A. Giani, and P. Thompson. Cognitive hacking: A battle for the mind. *Computer*, 35(8):50–56, 2002.

[12] D. Fetterly, M. Manasse, and M. Najork. Spam, damn spam, and statistics. In *WebDB2004*, June 2004.

[13] D. Fetterly, M. Manasse, M. Najork, and J. Wiener. A large-scale study of the evolution of web pages. In

Proceedings of the twelfth international conference on World Wide Web, pages 669–678. ACM Press, 2003.

[14] G. W. Flake, S. Lawrence, C. L. Giles, and F. Coetzee. Self-organization of the web and identification of communities. *IEEE Computer*, 35(3):66–71, 2002.

[15] I. for Propaganda Analysis. How to detect propaganda. *Propaganda Analysis*, 1(2), 1937.

[16] Google. The Google API. http://www.google.com/apis/.

[17] L. Graham and P. T. Metaxas. "Of course it's true; i saw it on the internet!": Critical thinking in the internet era. *Commun. ACM*, 46(5):70–75, 2003.

[18] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *WWW 2004*, May 2004.

[19] Z. Gyongui and H. Garcia-Molina. Web spam taxonomy. Technical Report TR 2004-25, Stanford University, 2004.

[20] Z. Gyongui, H. Garcia-Molina, and J. Pedersen. Combating web spam with TrustRank. In *VLDB 2004*, Aug. 2004.

[21] T. H. Haveliwala. Topic-sensitive pagerank. In *Proceedings of the eleventh international conference on World Wide Web*, pages 517–526. ACM Press, 2002.

[22] M. R. Henzinger. Hyperlink analysis for the web. *IEEE Internet Computing*, 5(1):45–50, 2001.

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

[24] M. Hindman, K. Tsioutsiouliklis, and J. Johnson. Googlearchy: How a few heavily-linked sites dominate politics on the web. In *Annual Meeting of the Midwest Political Science Association*, April 3-6 2003.

[25] L. Introna and H. Nissenbaum. Defining the web: The politics of search engines. *Computer*, 33(1):54–62, 2000.

[26] G. Jeh and J. Widom. Scaling personalized web search. In *Proceedings of the twelfth international conference on World Wide Web*, pages 271–279. ACM Press, 2003.

[27] J. Kleinberg. The small-world phenomenon: an algorithm perspective. In *STOC '00: Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 163–170. ACM Press, 2000.

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

[29] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the Web for emerging cyber-communities. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(11–16):1481–1493, 1999.

[30] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. The web and social networks. *IEEE Computer*, 35(11):32–36, 2002.

[31] A. M. Lee and E. B. Lee(eds.). *The Fine Art of Propaganda*. The Institute for Propaganda Analysis. Harcourt, Brace and Co., 1939.

[32] C. A. Lynch. When documents deceive: trust and provenance as new factors for information retrieval in a tangled web. *J. Am. Soc. Inf. Sci. Technol.*, 52(1):12–17, 2001.

[33] M. Marchiori. The quest for correct information on the web: hyper search engines. *Comput. Netw. ISDN Syst.*, 29(8-13):1225–1235, 1997.

[34] M. L. Maulding. Lycos: Design choices in an internet search service. *IEEE Expert*, January-February(12):8–11, 1997.

[35] G. Pringle, L. Allison, and D. L. Dowe. What is a tall poppy among web pages? In *Proceedings of the seventh international conference on World Wide Web 7*, pages 369–377. Elsevier Science Publishers B. V., 1998.

[36] P. Raghavan. Social networks: From the web to the enterprise. *IEEE Internet Computing*, 6(1):91–94, 2002.

[37] G. Salton. Dynamic document processing. *Commun. ACM*, 15(7):658–668, 1972.

[38] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, 1999.

[39] M. Totty and M. Mangalindan. As google becomes web's gatekeeper, sites fight to get in. In *Wall Street Journal CCXLI(39)*, February 26 2003.

[40] A. Vedder. Medical data, new information technologies and the need for normative principles other than privacy rules. In *Law and Medicine. M. Freeman and A. Lewis (Eds.), (Series Current Legal Issues)*, pages 441–459. Oxford University Press, 2000.

[41] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

[42] B. Wu and B. Davison. Identifying link farm spam pages. In *WWW 2005*, May 2005.

[43] yWorks. yEd – java graph editor, v. 2.2.1. http://www.yworks.com/en/products_yed_about.htm.