Undue Influence: Eliminating the Impact of Link Plagiarism on Web Search Rankings*

Baoning Wu   **Brian D. Davison**

Department of Computer Science & Engineering
Lehigh University, Bethlehem, PA 18015 USA
{baw4,davison}@cse.lehigh.edu

April 2006

Abstract

Link farm spam and replicated pages can greatly deteriorate link-based ranking algorithms like HITS. In order to identify and neutralize link farm spam and replicated pages, we look for sufficient material copied from one page to another. In particular, we focus on the use of "complete hyperlinks" to distinguish link targets by the anchor text used. We build and analyze the bipartite graph of documents and their complete hyperlinks to find pages that share anchor text and link targets. Link farms and replicated pages are identified in this process, permitting the influence of problematic links to be reduced in a weighted adjacency matrix. Experiments and user evaluation show significant improvement in the quality of results produced using HITS-like methods.

1 Introduction

The search engine is the door to the Web today. Many websites get significant traffic through search engine results, so it has become commercially expedient for a web site to rank as highly as possible. Search engine spamming is the use of techniques that provide an artificially high ranking, and is of significant concern to commercial search engines [24, 34].

Before the invention of link-based ranking algorithms, most search engine spam was related to content, and included techniques such as repeated key words, invisible text, long lists of descriptions of a page, etc. With the advent of link-based ranking algorithms such as PageRank [9] and HITS [25] in the search engine world, such textual manipulation became less effective, and so the focus turned to link-based techniques.

*Technical Report LU-CSE-06-007, Department of Computer Science and Engineering, Lehigh University, Bethlehem, PA, 18015. This report obsoletes LU-CSE-05-014.

When first introduced, algorithms like HITS (variants of which are found in search engines such as Teoma which powers Ask Jeeves) did quite well for fighting content spam and gave quite relevant results. However with the appearance of link farms, in which sites are densely interconnected, HITS is no longer robust [7, 27, 28, 11].

To demonstrate this, we gathered web pages for hundreds of queries and applied HITS to them in 2004. Manual review of those results indicated that more than 80 percent were distorted by spamming communities. For example, for the query *web proxy* served by Yahoo, the top ten authorities that result from HITS are listed in Table 1(a). Several of these sites are strongly connected with each other and they dominate the HITS authority results. Another example is for the query *weather* served by Google. The top ten HITS authorities are listed in Table 1(b), and most have nothing to do with weather. We will see more examples in Section 4.

Currently there are three major factors that can degrade HITS results:

- "Mutually reinforcing relationships" [7] in which a page is the target of many links from multiple pages of the same web site, or alternatively, one page that points to many pages of another web site.
- The existence of many duplicate pages, making links cited within them rank highly.
- Link farms [31, 22] in which a set of Web pages is densely interconnected.

Some of these issues have been addressed in part by other research, which we will discuss below in Section 2. Our approach is unique in that it addresses duplicate pages and link farms at the same time by examining "complete hyperlinks" to recognize repeated content.

The outline of our idea is as follows. We propose to use the concept of a "complete hyperlink", i.e., the link target together with its anchor text as a basic unit for each link. To make duplicate pages or build link farms, many links are intentionally created and duplicated. If we can identify these copied links, duplicate pages and link farms can be detected. Our intuition here is that duplication of a "complete link" is a much stronger sign of this copying behavior than just a duplicate link target. Based on this concept, we extract one or more "complete links" from each HTML document and build a document-hyperlink matrix for these pages. The bipartite subgraph within this matrix is a good indication of duplicate pages or link farms.

While we won't punish a page for a single copied hyperlink, if there are enough such duplicate complete links, the chance is small that all these links within this page are generated independently by different people. Therefore, we penalize such links. After finding these duplicate "complete links", we punish them by down-weighting them in the adjacency matrix and then calculate HITS results. Our policy is only to punish links, but not to ban pages containing these links. User evaluation of query results shows a significant improvement in the quality of results produced using HITS-like methods.

Rank	URL
1	http://linux.com/
2	http://newsforge.com/
3	http://itmj.com/
4	http://thinkgeek.com/
5	http://newsletters.osdn.com/
6	http://freshmeat.net/
7	http://slashdot.org/
8	http://www.ostg.com/
9	http://ads.osdn.com/?ad_id=2560&alloc_id=6182&op=click
10	http://ads.osdn.com/?ad_id=2435&alloc_id=5907&op=click

(a) Top HITS authorities for *web proxy* from Yahoo.

Rank	URL
1	http://www.tripadvisor.com/
2	http://www.virtualtourist.com/
3	http://www.abed.com/memoryfoam.html
4	http://www.abed.com/furniture.html
5	http://www.rental-car.us/
6	http://www.accommodation-specials.com/
7	http://www.lasikeyesurgery.com/
8	http://www.lasikeyesurgery.com/lasik-surgery.asp
9	http://mortgage-rate-refinancing.com/
10	http://mortgage-rate-refinancing.com/mortgage-calculator.html

(b) Top HITS authorities for *weather* from Google.

Table 1: HITS results dominated by link farms.

The remainder of this paper is organized as follows. Section 2 reviews related work about link farms, community discovery, and duplicate Web pages. The concept of "complete links" and the algorithm for finding the bipartite graph on the document-hyperlink matrix will be introduced in Section 3. Some experimental results and human evaluation results will be presented in Section 4. We wrap up with a discussion and summary of our conclusions.

2 Related work

The related work can be divided into three categories: one fights link farm spam, second generates clusters or to use bipartite core to find web communities, and the last finds duplicate web pages.

2.1 TKC effect and link farm spam

The problem of "mutually reinforcing relationships" is first introduced by Bharat and Henzinger [7]. A simple but effective algorithm is also given there, i.e., to assign the edge an authority weight of $1/k$ if there are k pages from the same host pointing to a single document on a second host and assign the edge a hub weight of $1/l$ if a single document on the first site has l links to a set of documents on a second host. Unfortunately, while this method neutralizes mutually reinforcing relationships, it cannot handle larger groups acting in cohort.

The "TKC effect" is first described by Lempel and Moran [27]. Pages within a tightly-knit community will get high rank value for iterative processes like HITS. A link farm exploits the TKC effect to improve their position in search engine rankings. The authors propose the SALSA algorithm which is more resistant to the TKC effect than HITS. Unfortunately, SALSA does not incorporate long-range reinforcement that HITS has (rather it acts more like a popularity ranking method [12, 8]), and is still vulnerable to a form of the TKC effect when many pages all point to each other [33].

Chakrabarti proposed using Document Object Models together with hyperlinks to beat nepotistic "clique attacks" [11]. Compared to his idea, our method only takes the link and simple page content (i.e., anchor text) into account and doesn't need any other content analysis.

Li et al. [28] found that HITS is vulnerable to the "small-in-large-out" situation. The "small-in-large-out" is the root link that has few in-links but a large number of out-links. Usually the community associated with this root link will dominate the HITS results. They used a revised HITS algorithm that will give these "small-in-large-out" root links different weights to ameliorate their effects. Since their approach are not aimed at link farm spam, it is quite possible that they may still fail under the attack of link farms.

Gyongyi et al. describe a new algorithm, TrustRank, to combat Web spam [23]. The basic assumption of TrustRank is that good pages usually point to good pages and seldom have links to spam pages. They first select a bunch of known good seed pages and assign high trust scores to them. They then follow an approach similar to PageRank; the trust score is propagated via out-links to other Web pages. Finally, after convergence, the pages with high trust scores are believed to be good pages. Their algorithm is not appropriate to Web island pages.

Recently, Benczur et al. [4] proposed SpamRank to find spam pages by finding pages whose inlink pages formed deviant distributions of PageRank, marking them as likely spam, and then propagating the spam weight using PageRank on the inverted web graph. Thus, at the end, each page in the graph has a SpamRank correlated to how close it is to believed spam pages.

Zhang et al. show how to use the "amplification factor" in PageRank results to detect link collusions [39]. They define a new metric "amplification factor" as the total weight of PageRank value of a group of nodes over the actual weight of PageRank value flowing into this group. The intuition behind "amplification factor" is that the stationary weight of colluding nodes to be highly correlated with the dumping factor while non-colluding nodes to be relatively insensitive. By using different dumping factors and drawing the correlation curve for each page, they can identify most of the colluding nodes.

Fetterly et al. use statistical analysis to find spam [18]. Several distribution graphs, such as the distribution of the number of different host-names mapping to the same IP address, or the distribution of out-degrees, are drawn. Most of these distributions are modeled well by formulas such as a power law. The outliers of such datasets are marked as spam candidates. By manually checking these candidates, a majority of them are found to be spam.

Amitay et al. [2] propose categorization algorithms to detect website functionality. They recognized that sites with similar roles exhibit similar structural patterns. In the experiment results, they claimed to have identified 31 clusters, each of which appears to be a spam ring.

Elsewhere, we present a different method for finding link farm spam [36], in which an automated method is used to select a seed set of bad pages (spam pages) which are then expanded to include others that link to enough known bad pages. Our current approach is additionally able to find duplicated pages and has an accuracy of about 5% higher.

Finally, in prior work [15] we examined the viability of recognizing whether a hyperlink would be considered nepotistic. Using a variety of page characteristics, initial experimental results on a small dataset showed significant promise, achieving classification accuracies of over 90%.

2.2 Web communities or clusters

Kumar et al. [26] used bipartite subgraphs and cores to find web communities. Figure 1 is an example of a complete bipartite graph. They use the Apriori algorithm to find all bipartite cores and then expand the core to communities by using HITS-like methods. In order to prevent duplicate hubs from identifying cores as spurious communities, they filter hubs that contain duplicate link shingles — defined as the hashes of a sequence of five links.

Eppstein [16] shows a linear time algorithm to list all the maximal complete bipartite subgraphs. The work is interesting but it requires that the graph must satisfy bounded arboricity and requires to be implemented in a parallel processing environment. Also, Zaki and Ogihara [38] mention that Eppstein's algorithm is not practical for large data sets due to the big constant within the time complexity formula.

Flake et al. [19] defines a web community as "a set of sites that have more links (in either direction) to members of the community than to non-members." They proposed using maximum flow and minimum cut method to identify these communities. They only take the link graph into account, while our method utilize anchor text together with links.

Reddy and Kitsuregawa also used bipartite graphs to find web communities [32]. Unlike Kumar et al., they are not limited to complete bipartite graphs, but

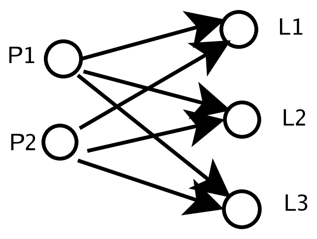


Figure 1: A complete bipartite graph with size 2*3.

instead propose a method to find densely connected bipartite graphs to identify larger web communities.

Roberts and Rosenthal [33] propose a simple algorithms to find clusters of web pages based on their outlink sets, and the authority value of a page is proportional to the number of clusters which tend to link to it rather than the number of pages which link to it.

2.3 Duplicate and near-duplicate Web pages

A number of significant research papers about detecting duplicates of web pages have been published [10, 5, 35, 13, 6, 17]. The basic idea of these papers is to first cut the documents into chunks or shingles with a sequence of terms. They then either use these chunks directly or calculate a digest of the chunks as the fingerprint of each document. Two documents are considered duplicates if they have significant overlap of the chunks or fingerprints. In our approach, only links and anchor texts are extracted from a page and two pages are considered to contain duplicate material if they have common links with common anchor texts without requiring them to be in sequence. We will give details in Section 3.

Some researchers have also examined the use of templates in Web pages. Bar-Yossef and Rajagopalan [3] used DOM trees to extract pagelets, which can be used to form templates, from Web pages. Then they ran the pagelet/template-based Clever to find that precision is increased significantly. Gibson et al. [20] also used both DOM based and text based algorithm to detect templates, and demonstrated an increasing trend of templates on the Web. While our approach is somewhat similar to template recognition, our matching is less brittle to content and link variations created by spammers.

3 Algorithm details

In this section, we give details of our algorithm for addressing link farm and duplicate page issues, and can be summarized as follows:

1. Collect pages for a given query by using the HITS collection process.

2. Parse all pages to get “complete hyperlinks” for each page and generate a document-hyperlink matrix. Filter out hyperlinks to pages within the same domain.
3. Find densely connected bipartite components within the matrix generated in step 2, by identifying the union of all satisfying complete bipartite graphs.
4. Change the adjacency matrix with weighted values according to the components found in step 3.
5. Apply additional re-weighting such that l links from one domain to a single page to have weight $1/l$ (identical to one half of Bharat and Henzinger’s *imp* improvement to HITS [7]).
6. Calculate final ranking using re-weighted link graph.

3.1 Complete hyperlinks

In our algorithm we propose to use “complete hyperlinks”, i.e., to consider one hyperlink together with its anchor text as a single unit. The “complete link” is used instead of just the link target as in previous link analysis work by others. If two pages have a bunch of common links with same anchor texts, it is a stronger sign than using links alone that these two pages are duplicates or made by the same person or machine on purpose, which is an obvious behavior of link farm spammers.

Since a “complete link” is the combination of link and its anchor text, it is quite possible that two different complete links will have same URL but with different anchor texts. For example, the following tuples are two different complete links:

(`http://wume.cse.lehigh.edu/`, “WUME lab”)

(`http://wume.cse.lehigh.edu/`, “Our lab’s homepage”)

In the parsing process, for each page we get all the outgoing links together with their anchor texts. Suppose we have a data set of n pages for a query and after parsing we get together m different complete hyperlinks. Then for this data set, an $n * m$ size “document-hyperlink” matrix A can be built. If the document i contains complete link j, then $A[i, j]$ will be set to 1, otherwise 0.

3.2 Finding bipartite components

A bipartite graph has two disjoint sets X and Y. In a complete bipartite graph, each element in X will point to each element in Y and each element in Y is pointed to by each element in X. If there are k elements in set X and l elements in set Y, the bipartite graph has size $k * l$.

For the “document-hyperlink” matrix $A[n * m]$ generated above, we can find bipartite components with the set X filled with documents and set Y filled with complete links. If the bipartite component sizes are above some thresholds, we will consider them as a problematic community. The reason is that there are two possibilities that real pages will form a large bipartite component within the document-hyperlink matrix: one is a link farm and the other is duplicate pages. For example, if we set the thresholds to be 10 and 5, that means we will find all bipartite components that have at least 5 documents and each of them have 10 duplicate complete hyperlinks with others. It is quite likely that these 5 pages are duplicates or they are connected as a result of some kind of collusion.

In reality, link farms often do not form a single complete bipartite component; instead we have found that they are usually densely connected via multiple overlapping small bipartite cores.

One thing to note here is that to list all the complete bipartite components is not an easy job. Here, we propose an approximate algorithm to list complete bipartite components from a given "document-hyperlink" matrix. This approximate algorithm is used in our paper [37].

3.2.1 An approximate Algorithm

Initial Steps. It is not necessary to find each overlapping core; our algorithm aims at finding the pages forming complete bipartite cores with at least k documents, each containing at least l common complete links where k and l are thresholds that can be adjusted.

The basic idea of our algorithm is that we don't consider each bipartite component alone, that is, we only consider the document-hyperlink matrix as a whole and each time we only delete the rows or columns which don't have enough non-zero members according to the thresholds. This process will iterate until no more rows or columns can be deleted from the matrix. For a simple example, a starting matrix with 5 documents and 4 complete links is given in Figure 3(a), and the thresholds are 2 and 2. We need to find all bipartite components which have at least 2 documents and each of them has at least 2 common complete links. Once the iterations are complete, we will get the matrix in Figure 3(b).

One problem with the above step is that sometimes an element will be marked incorrectly in the final matrix because it just has enough links within another bipartite component. As we can see from Figure 3(b) that the $A[P4, L1]$ is still 1 because P1 and P2 have enough number of L1, but we don't want to punish the link L1 in page P4. So in order to get the final correct matrix as Figure 3(c), we need a final adjusting step. A summary of the process of finding bipartite components is shown in Figure 2.

Final Step. Given document-hyperlink matrix A, we calculate a special form of $B = A * A^T$ to get a document-document matrix in which $B[i, j]$ records the common hyperlinks within document i and document j. For example, if the given matrix is that in Figure 3(b), we will get the matrix in Table 2 after this step.

For each $B[i, j]$, if the number of different members is less than the threshold l or $i = j$, then the element is removed. For example, in Table 2, P1 and P2 have two members, which is equal to the threshold 2. So the link P1-L1 and

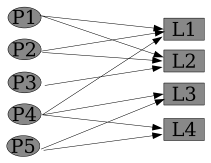
Finding Bipartite Components:

INPUT: document-hyperlink matrix A; thresholds k and l
OUTPUT: reduced document-hyperlink matrix with only bipartite components of $k * l$ or larger

1. Zero all rows of A that have a sum smaller than l
2. Zero all columns of A that have a sum smaller than k
3. Repeat Steps 1 and 2 until A no longer changes.

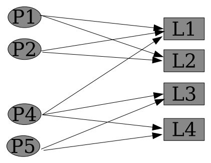
Figure 2: Algorithm for finding bipartite components.

	L1	**L2**	**L3**	**L4**
P1	1	1	0	0
P2	1	1	0	0
P3	0	1	0	0
P4	1	0	1	1
P5	0	0	1	1



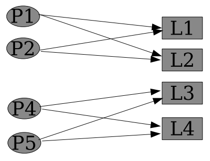
(a) Initial document-hyperlink matrix.

	L1	**L2**	**L3**	**L4**
P1	1	1	0	0
P2	1	1	0	0
P4	1	0	1	1
P5	0	0	1	1



(b) After one iteration.

	L1	**L2**	**L3**	**L4**
P1	1	1	0	0
P2	1	1	0	0
P4	0	0	1	1
P5	0	0	1	1



(c) Final document-hyperlink matrix.

Figure 3: Document-hyperlink matrix views.

P1-L2 are valid. But for row of P4, P4 and P1 have only 1 common complete link L1, so the link P4-L1 is removed from $B[P4, P1]$ and $B[P4, P2]$.

Remove all elements in matrix A that are not present in B. For example,

Final Adjustment:

INPUT: Partially reduced document-hyperlink matrix A
OUTPUT: Final reduced document-hyperlink matrix A

- Generate a matrix B in which $B[i, j]$ contains the intersection of the complete hyperlinks of documents i and j, as long as as $i \neq j$.
- For each element $B[i, j]$, drop the element if the number of complete links is less than the threshold l.
- Remove all elements in A that are not present in B.

Figure 4: Detail of the final adjusting step.

	P1	**P2**	**P4**	**P5**
P1	0	L1,L2	L1	0
P2	L1,L2	0	L1	0
P4	L1	L1	0	L3,L4
P5	0	0	L3,L4	0

Table 2: Document-document matrix B.

since P4-L1 is no longer present, that element will be set to 0. Then we get the matrix in Figure 3(c).

The final adjusting algorithm is shown in Figure 4. Now all the bipartite components are stored in the matrix A and in order to distinguish this new matrix from the original document-hyperlink matrix, we call this new matrix the bipartite matrix.

Computational Complexity The most computationally expensive part of our algorithm is to find bipartite subgraphs for the document-hyperlink matrix. There are 4 steps for finding the bipartite components of a matrix. The time complexity of Steps 1 and 2 are $O(n * m)$ each; Step 3 loops over Steps 1 and 2 at most 2*min(n,m) times. So the time complexity of the first three steps are $O(nm{*}\min(n,m))$. The final step has a complexity of $O(n^2m)$. So if $n = m$, in total the time complexity for detecting bipartite components is $O(n^3)$.

3.2.2 A simpler alternative

The above algorithm has flaws when the threshold for k is bigger than 2. The reason is that the final adjusting step will keep all the common links that appear in two different documents. For example, if we consider the following example matrix in Table 3 and use 3 as the threshold for both k and l, then the result matrix will remain unchanged after applying the whole algorithm in Section 3.2.1. Obviously the link P7-L3 and P8-L3 should be dropped because they do not match the threshold 3. The reason that our algorithm fails to drop them

	L1	**L2**	**L3**	**L4**	**L5**	**L6**
P1	1	1	1	0	0	0
P2	1	1	1	0	0	0
P3	1	1	1	0	0	0
P4	0	0	0	1	1	1
P5	0	0	0	1	1	1
P6	0	0	0	1	1	1
P7	0	0	1	1	1	1
P8	0	0	1	1	1	1

Table 3: A document-hyperlink matrix that the algorithm can not handle correctly.

is that the intersection of P7 and P8 is [L3,L4,L5,L6], thus four elements exist in the corresponding document-document matrix. Since four is bigger than the threshold three, all these four elements will be kept in the final Document-hyperlink matrix according to the final adjusting step shown in Figure 4.

This indicates that our proposed algorithm may penalize some link farm pages more than enough. Since the document-document matrix can guarantee that all the marked complete links occur in at least two documents, thus, if we fix 2 as the value for k, then the above flaw will not be a problem any more. Hence, we propose using this fixed value 2 for k as an alternative for the bipartite core detecting.

By fixing k, as long as more than l common complete links exist in two documents, these links will be marked as link farm links. Hence, the threshold for l should be increased such that only link farms will be penalized. For example, if two documents share 4 or 5 common complete links, these two documents may not be a problem. But if two documents share more than 10 common complete links, the chance is big that these links are within a same link farm.

Another advantage of this simpler alternative is that the speed of detecting bipartite components is improved. Because one direct implementation of this alternative can just compare each document with all the others. Hence the complexity will be $O(n^2)$ for this simpler alternative.

3.3 Down-weighting the adjacency matrix

As we mentioned before, there are two possible methods to form the bipartite core, one is the link farm and the other is duplicate pages. To eliminate their effects on link-based ranking algorithms, one intuition is to give lower weight to these links within the adjacency matrix as in [7, 33].

Based on the bipartite matrix, a straightforward way to down-weight the adjacency matrix is used in our algorithm: For each unique complete link L, we count the total N appearances of L in the bipartite matrix. Then for each appearance of L in the final document-hyperlink matrix, we assign $1/N$ in the adjacency matrix instead of 1, which is the default value.

For example, in Figure 3(c), $L1$ appears twice, so both $A[P1, L1]$ and

	L1	**L2**	**L3**	**L4**
P1	0.5	0.5	0	0
P2	0.5	0.5	0	0
P3	0	1	0	0
P4	1	0	0.5	0.5
P5	0	0	0.5	0.5

Table 4: Down-weighted document-hyperlink matrix.

$A[P2, L1]$ in Figure 3(a) will be set to 1/2. The final down-weighted matrix for this example is shown in Table 4.

3.4 Ranking using the weighted adjacency matrix

The final step is to use the down-weighted adjacency matrix for ranking the results. Several methods can be used in this step, and we tried both Kleinberg's HITS and ranking by popularity.

First we used original HITS on this weighted adjacency matrix and found that the results are not good for some queries. By inspection, we find that the main reason is that "mutually reinforcing relationships" exist. So, inspired by Bharat and Henzinger's *imp* version of HITS, we re-weight instances of l links from one domain to a single page to have weight $1/l$ to address this problem. After solving this problem, we find that most tested queries get good results.

The second algorithm is just ranking by popularity. Here "popularity" of a page means the sum of all weighted values for the incoming links to the page. This algorithm is fast because no iterative process is involved and usually it can also generate good ranking results.

4 Experiments and evaluation

In this section, we will describe the data sets that are used in our experiments, the ranking results of our algorithms discussed in the previous sections, and finally a user study to evaluate our results.

In this experimental section, without specific indication, the results are calculated by the algorithm introduced in Section 3.2.1. In general, we use 5*5 as the thresholds for k and l.

4.1 Data set

We adopted the same data collecting process as HITS to collect our experiment data. Initially we send a query to search.yahoo.com get top 200 URLs for this query, then for each URL, we get the top 50 incoming links to this URL by querying search.yahoo.com again and we also download all outgoing pages within this URL. The expanded data set is then used as the base data set for this query.

Rank	URL
1	http://www.teleactivities.org/
2	http://www.no-gambling.com/
3	http://www.danubecasino.com/
4	http://www.teleorg.org/
5	http://www.teleactivities.com/
6	http://www.teleactivities.net/
7	http://www.teleactivities.net/resources/ezines.htm
8	http://www.teleactivities.net/resources/sites.htm
9	http://www.teleactivities.net/resources/forums.htm
10	http://www.teleactivities.net/resources/awards.htm

(a) HITS results.

Rank	URL
1	http://www.no-gambling.com/
2	http://www.teleorg.org/
3	http://ong.altervista.org/
4	http://bx.b0x.com/
5	http://video-poker.batcave.net/
6	http://www.websamba.com/marketing-campaigns
7	http://online-casino.o-f.com/
8	http://caribbean-poker.webxis.com/
9	http://roulette.zomi.net/
10	http://teleservices.netfirms.com/

(b) BH-HITS results.

Rank	URL
1	http://www.freetranslation.com/
2	http://www.systransoft.com/
3	http://babelfish.altavista.com/
4	http://www.yourdictionary.com/
5	http://dictionaries.travlang.com/
6	http://www.google.com/
7	http://www.foreignword.com/
8	http://www.babylon.com/
9	http://www.worldlingo.com/products_services/worldlingo_translator.html
10	http://www.allwords.com/

(c) CL-HITS and CL-POP results.

Table 5: Results for query *translation online*.

Rank	**URL**
1	http://www.recallusa.com/
2	http://www.binarybiz.com/
3	http://www.medi-spastore.com/
4	http://www.101domain.com/
5	http://www.lencom.com/
6	http://www.lastminute-angebote.ws/
7	http://www.toploansources.com/debt-consolidation.htm
8	http://www.losangeleshairstudio.com/los_angeles_hair_salon
9	http://www.credit-rocket.com/
10	http://rwgusa.com/advertise.htm

(a) HITS results.

Table 6: Initial results for query *auto shopping*.

For this experiment, we used 412 queries and downloaded more than 2.1 million unique Web pages. These queries include queries used by previous researchers [25, 33, 11, 14, 27, 28], the names of the categories from the dmoz Open Directory Project [30], and popular queries from Lycos and Google [1, 21].

4.2 Experimental results

For each query, we compare the rankings generated by four methods. The first is simply HITS on the original graph. The second is BH-HITS (inspired by Bharat and Henzinger's *imp*), which is HITS applied to the graph after re-weighting l links from one domain to a single page to have weight $1/l$. The third is CL-HITS, which is HITS applied to the BH-HITS weighted graph but also incorporates re-weightings based on bipartite components found, and the last is CL-POP, which applies weighted popularity to rank pages on the same graph as CL-HITS.

In most cases the HITS will generate bad results and sometimes BH-HITS will be dominated by link farms. In contrast, CL-HITS and CL-POP usually give good results. In the following, we provide experimental results for some sample queries.

Query: translation online. The top 10 authority URLs for this query by HITS are in Table 5(a). The top 10 authority URLs for this query by BH-HITS are in Table 5(b). The top 10 authority URLs for this query by CL-HITS and CL-POP are identical, and are shown in Table 5(c).

From these tables, we can see that HITS is dominated by pages from `http://www.teleactivities.com/`, where a lot of nepotistic links [26, 15] exist. BH-HITS solves this problem, but unfortunately, is still dominated by several strongly connected pages, which has nothing to do with the query *translation online*. The CL-HITS and CL-POP do quite well with same top lists, but CL-POP needs significantly less computation.

Rank	**URL**
1	http://www.jahanshahr.com/
2	http://www.creativenative.net/
3	http://www.college-school-loans.com/
4	http://www.consolidation-loans-finder.net/
5	http://www.1314168.com/
6	http://www.cybernetixusa.com/
7	http://www.achclan.com/
8	http://www.college-student-loans.org/
9	http://www.ccrowsfansart.com/
10	http://www.mobile-home-loans-home-equity-loans.com/index.html

(a) BH-HITS results.

Rank	**URL**
1	http://www.autopartsreplacement.com/
2	http://www.paintscratch.com/
3	http://www.autobodypartsonline.com/
4	http://www.autobytel.com/
5	http://www.autodirectsave.com/
6	http://www.autozone.com/
7	http://www.buyautoparts.com/
8	http://www.edelbrock.com/
9	http://www.jcwhitney.com/
10	http://www.holley.com/

(b) CL-HITS results.

Rank	**URL**
1	http://www.autopartsreplacement.com/
2	http://www.carsale.com/
3	http://www.autobytel.com/
4	http://www.paintscratch.com/
5	http://www.auto-rx.com/
6	http://www.autozone.com/
7	http://www.autopartslive.com/
8	http://www.autopartstreet.com/
9	http://www.autobodypartsonline.com/
10	http://www.autopartsauthority.com/

(c) CL-POP results.

Table 7: Results for query *auto shopping*.

Rank	URL
1	http://www.discountcars.net/
2	http://www.motel-discounts.com/
3	http://www.stlouishoteldeals.com/
4	http://www.richmondhoteldeals.com/
5	http://www.jacksonvillehoteldeals.com/
6	http://www.jacksonhoteldeals.com/
7	http://www.keywesthoteldeals.com/
8	http://www.austinhoteldeals.com/
9	http://www.gatlinburghoteldeals.com/
10	http://www.ashevillehoteldeals.com/

(a) HITS results.

Table 8: Initial results for query *rental car*.

Query: auto shopping. The top 10 authority URLs for this query by HITS are in Table 6(a). The top 10 authority URLs for this query by BH-HITS are in Table 7(a). The top 10 authority URLs for this query by CL-HITS are in Table 7(b). The top 10 authority URLs for this query by CL-POP are in Table 7(c).

Again, HITS is dominated by nepotistic links of pages from `www.bethpagechamber.com`; the top 10 URLs from BH-HITS are densely connected to each other. CL-HITS and CL-POP both give relevant top results; there are some different URLs within top 10 for these two ranking methods, but in fact in our whole list, the top 30 URLs of them are similar but with different order.

Query: rental car. The top results for the four algorithms are in Tables 8(a)-9(c). HITS and BH-HITS are both overwhelmed with link farms. CL-HITS and CL-POP both give relevant results, with the exception of `google.com` in position 5 for CL-POP. Google shows up in position 15 for CL-HITS.

4.3 Duplicate pages

Duplicate pages can be a problem for Web page ranking. One common approach is to delete these duplicate pages before ranking [26, 32].

There are some difficulties with this deleting process. The most obvious question is which duplicate to keep and which one to delete. So we believe it is better to give weighted value to the outgoing links within duplicate pages instead of keeping one page and deleting the rest. Another problem is that two pages may escape from the shingle test [10] while in fact a lot of links within these two pages are common while much text content of these two pages are different. By giving different weights, only duplicate links within duplicate pages are punished; other unique links within these duplicate pages are still retain their original value.

Rank	**URL**
1	http://www.rentadeal.com/
2	http://www.allaboutstlouis.com/
3	http://www.allaboutboston.com/
4	https://travel2.securesites.com/about_travelguides/addlisting.html
5	http://www.allaboutsanfranciscoca.com/
6	http://www.allaboutwashingtondc.com/
7	http://www.allaboutalbuquerque.com/
8	http://www.allabout-losangeles.com/
9	http://www.allabout-denver.com/
10	http://www.allabout-chicago.com/

(a) BH-HITS results.

Rank	**URL**
1	http://www.hertz.com/
2	http://www.avis.com/
3	http://www.nationalcar.com/
4	http://www.thrifty.com/
5	http://www.dollar.com/
6	http://www.alamo.com/
7	http://www.budget.com/
8	http://www.enterprise.com/
9	http://www.budgetrentacar.com/
10	http://www.europcar.com/

(b) CL-HITS results.

Rank	**URL**
1	http://www.hertz.com/
2	http://www.avis.com/
3	http://www.thrifty.com/
4	http://www.nationalcar.com/
5	http://www.google.com/
6	http://www.alamo.com/
7	http://www.dollar.com/
8	http://www.budget.com/
9	http://www.bnm.com/
10	http://www.enterprise.com/

(c) CL-POP results.

Table 9: Results for query *rental car*.

Rank	**URL**
1	http://www.maps.com/
2	http://www.mapsworldwide.com/
3	http://www.cartographic.com/
4	http://www.amaps.com/
5	http://www.cdmaps.com/
6	http://www.ewpnet.com/maps.htm
7	http://mapsguidesandmore.com/
8	http://www.njdiningguide.com/maps.html
9	http://www.stanfords.co.uk/
10	http://www.delorme.com/

(a) BH-HITS results.

Rank	**URL**
1	http://www.maps.com/
2	http://maps.yahoo.com/
3	http://www.delorme.com/
4	http://tiger.census.gov/
5	http://www.davidrumsey.com/
6	http://memory.loc.gov/ammem/ gmdhtml/gmdhome.html
7	http://www.esri.com/
8	http://www.maptech.com/
9	http://www.streetmap.co.uk/
10	http://www.libs.uga.edu/darchive /hargrett/maps/maps.html

(b) CL-HITS results.

Table 10: Results for query *maps*.

As mentioned before, our bipartite component detection algorithm can detect duplicate links among pages. Since we can still use the same mechanism for weighting as described in section 3.3, no preprocessing step is necessary to detect these duplicate pages.

For example, for the query *maps*, people usually want to find web pages related to maps, driving directions, etc. The top 10 authority URLS for this query by BH-HITS are in Table 10(a). Although most of the top URLs are related to maps, several more famous websites for map don't show up here, such as `maps.yahoo.com`. The reason for this is there are quite some duplicate pages that are the replication of `http://dmoz.org/Shopping/Publications/Maps/`, but `maps.yahoo.com` is not within this page.

Our bipartite component detecting algorithm finds this automatically, so the weights for these pages are reduced. The result of CL-HITS for maps is in Table

	HITS	BH-HITS	CL-HITS	CL-POP
1	12.9%	24.5%	48.4%	46.3%
2	10.7%	18.3%	28.8%	26.2%
3	6.6%	10.4%	6.7%	6.4%
4	26.8%	14.7%	11.3%	12.7%
5	42.8%	31.9%	4.6%	8.1%

Table 11: Evaluation results.

10(b). We can see the results have more diversity than BH-HITS. So BH-HITS itself is not good at handling replicated pages. Our algorithm can achieve this without a separate process.

4.4 Evaluation of rankings

To compare the relevance of results generated by our algorithms with the results generated by HITS and BH-HITS, an evaluation web interface was built.

For this evaluation, 20 queries were selected. For each query, the top 10 URLs generated from 4 algorithms HITS, BH-HITS, CL-HITS and CL-POP were mixed for a blind evaluation. Users were presented with a randomly chosen query and a randomly selected set of ten results (in random order). For each URL presented with a query, study participants had five choices of relevance: 1. quite relevant, 2. relevant, 3. not sure, 4. not relevant, and 5. totally irrelevant for each URL. Only one choice was permitted for each URL (although a user could refrain from making a choice). After completion of each set of evaluations, a participant could optionally choose to evaluate another set of results for another randomly selected query.

The study included twenty-eight participants, recording a total of 1688 preferences. The distribution of preference scores is shown in Table 11. Both HITS and BH-HITS have bigger percentages in "not relevant" and "totally irrelevant", while CL-HITS and CL-POP have bigger percentages in "quite relevant" and "relevant". The major reason of this big difference is that we have much less spam results by using CL-HITS and CL-POP.

We can also use this data to calculate precision at 10, if we allow either a "quite relevant" or "relevant" choice to count. Under that assumption, HITS achieved a precision at rank 10 of 23.6%, BH-HITS 42.8%, CL-HITS 77.2%, and CL-POP 72.5%.

4.5 Results for the alternative method

As described in Section 3.2.2, we can fix the threshold for k to be 2. In choosing the threshold for l to be 10, we calculated the results again by using this alternative method. The results are shown in Table 12.

From Table 12, we can see that by using the alternative method, CL-HITS achieves a precision at rank 10 of 75.4% and CL-POP achieves a precision at rank 10 of 72.7%. Compared with the results shown in Table 11, this simple

	CL-HITS	**CL-POP**
1	45.9%	46.1%
2	29.6%	26.6%
3	5.8%	5.9%
4	11.4%	12.7%
5	6.9%	8.4%

Table 12: Evaluation results for the alternative method.

alternative can generate comparable results. Also the experimental results indicate that although a flaw exists in the approximate algorithm shown in Section 3.2.1, the results from the approximate method are slightly better than the results from the alternative method discussed in Section 3.2.2.

5 Discussion

In this paper, we have demonstrated the effectiveness of our algorithm in improving the quality of HITS-style results despite the presence of significant link spam. We made the explicit decision to penalize spamming behavior by re-weighting the graph, and did not attempt to penalize specific pages. In many cases, we find bipartite graphs of popular, high-quality interlinked sites that are under common corporate ownership. We chose to retain such pages (as well as more obviously spamming pages) in our dataset, but to reduce the effect that the bipartite component will have on the overall computation. In this section, we reconsider our choice of complete links as we have defined them above.

5.1 Complete link vs. link target

Intuitively, more link farms may be found by using link targets than complete links, because more targets will match and thus more easily exceed the bipartite component threshold than complete links; it is therefore possible that we may miss some link farms by requiring complete links.

On the other hand, the use of link targets alone can be too aggressive. Good links will end up down-weighted if these links are above the threshold for detecting bipartite components. For example, for the query *amusement parks*, the top result is `http://www.pki.com/` for the complete links method. This web site is a good one for this query, but we can't find it within top 30 list by using link targets alone because it is marked in one of the bipartite components and thus many incoming links get down-weighted.

To compare the performance of using the links alone with the performance of using complete links, we selected the same 20 queries, and for each query we applied the same algorithm but using links without anchor texts. Users evaluated the top 10 authorities for these queries. The performance is shown in Table 13. As we can tell, the sum of the "quite relevant" and "relevant" is 66.4%, which is about 10% lower than the results of using complete links,

Type	Results
quite relevant	44.3%
relevant	22.1%
not sure	5.4%
irrelevant	14.3%
totally irrelevant	13.6%

Table 13: Evaluation results for using links alone.

showing that using complete links is better than using link alone.

5.2 Exact match vs. bag of words

One obvious question people may raise is that what if the spammers deliberately use different anchor texts for the same link. Since our algorithm counts same link with different anchor texts as different complete links, the deliberately generated anchor texts for the same link may escape our bipartite graph detection and more spam links may survive.

In order to show the robustness of our algorithm to this issue, we did one more experiment to find the impact of the different anchor texts. In this experiment, instead of using the exact match of the anchor text, we used the idea of "bag of words" for handling the anchor texts, i.e., for each anchor text, we removed stop words and stemmed each word and put them in alphabetical order. Then this bag of words is used as the anchor text and forms the complete link together with the corresponding link. Similar algorithms are used for this new complete link and 20 queries are calculated for users evaluation. The results are shown in Table 14.

The sum of "quite relevant" and "relevant" from Table 14 is 76.2%, which is quite close to the results of using exact anchor text matching. This suggests that either exact anchor text matching or bag of words matching could be used.

Although one may think that spammers can use randomly generated anchor texts for the same link target in order to bypass our proposed method, we argue that in reality this won't work as they expect. The reason is that usually the spammers not only want to boost the link popularity to a page but also want to boost the page ranking for certain queries. In order to do this, they effectively combine link farms with link bombing techniques [29], i.e., they usually use common anchor texts to the links within a link farm.

As a result, our approach (especially the bag of words version) is expected to be an effective deterrent when its use is public knowledge. Spammers will have to expend more effort (generating different anchor texts to escape detection) and will have a less effective boost in rank (since the link bombing effect will be reduced by the use of different anchors).

Finally, we note that some of the occasional obvious errors in retrieval are the result of ranking without any additional textual analysis. Highly authoritative sites such as `google.com` will sometimes be listed in the top results, even when the query is unrelated. This problem may be corrected using techniques such

Type	Results
quite relevant	47.3%
relevant	28.9%
not sure	7.7%
irrelevant	11.4%
totally irrelevant	4.5%

Table 14: Evaluation results for using bag of words.

as pruning of outliers [7].

6 Conclusion

In this paper, we proposed a polynomial time method using complete links for detecting link farms and duplicate pages in order to get better results for link-based ranking algorithms. In addition, we propose a simpler alternative method for the same purpose. Both of them show big improvements when comparing with HITS algorithm.

While we cannot claim that our approach complete addresses the problem of search engine spam, we found that it can identify highly authoritative and relevant results, showed significant improvements over the original HITS performance on current, spam-dominated datasets, and for its effectiveness as a deterrent even if use became public knowledge. We also found that after graph re-weighting, ranking by popularity often provides similar quality to HITS at a much lower computational cost.

Acknowledgments

We thank the twenty-eight anonymous volunteers of our query relevance study.

References

[1] Lycos 50, 2005. http://50.lycos.com/.

[2] E. Amitay, D. Carmel, A. Darlow, R. Lempel, and A. Soffer. The connectivity sonar: Detecting site functionality by structural patterns. In *Proceedings of the Fourteenth ACM Conference on Hypertext and Hypermedia*, pages 38–47, Aug 2003.

[3] Z. Bar-Yossef and S. RajagopalanZiv. Template detection via data mining and its applications. In *WWW 2002, Honolulu, Hawaii, USA*, May 2002.

[4] A. A. Benczur, K. Csalogany, T. Sarlos, and M. Uher. SpamRank - fully automatic link spam detection. In *Proceedings of the First International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2005.

[5] K. Bharat and A. Broder. Mirror, mirror on the web: a study of host pairs with replicated content. In *Proceedings of the Eighth International World Wide Web Conference*, May 1999.

[6] 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.

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

[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 10th World Wide Web conference*, pages 415–429, May 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] A. Broder, S. Glassman, M. Manasse, and G. Zweig. Syntactic clustering of the web. In *Proceedings of the Sixth International World Wide Web Conference*, pages 1157–1166, 1997.

[11] S. Chakrabarti. Integrating the document object model with hyperlinks for enhanced topic distillation and information extraction. In *Proceedings of 10th World Wide Web conference*, pages 211–220, 2001.

[12] S. Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan Kaufmann, San Francisco, CA, 2003.

[13] J. Cho, N. Shivakumar, and H. Garcia-Molina. Finding replicated Web collections. pages 355–366, 2000.

[14] D. Cohn and H. Chang. Learning to probabilistically identify authoritative documents. In *Proc. 17th International Conf. on Machine Learning*, pages 167–174. Morgan Kaufmann, San Francisco, CA, 2000.

[15] B. D. Davison. Recognizing nepotistic links on the Web. In *Artificial Intelligence for Web Search*, pages 23–28. AAAI Press, July 2000. Presented at the AAAI-2000 workshop on Artificial Intelligence for Web Search, Technical Report WS-00-01.

[16] D. Eppstein. Arboricity and bipartite subgraph listing algorithms. *Information Processing Letters*, 51(4):207–211, 1994.

[17] D. Fetterly, M. Manasse, and M. Najork. the evolution of clusters of near-duplicate web pages. In *Proceedings of 1st Latin American Web Congress*, Nov 2003.

[18] D. Fetterly, M. Manasse, and M. Najork. Spam, damn spam, and statistics: Using statistical analysis to locate spam web pages. In *Proceedings of WebDB*, pages 1–6, June 2004.

[19] G. Flake, S. Lawrence, and C. L. Giles. Efficient identification of web communities. In *Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 150–160, Boston, MA, August 20–23 2000.

[20] D. Gibson, K. Punera, and A. Tomkins. The volume and evolution of web page templates. In *Proceedings of the 14th International World Wide Web Conference, Industrial and Practical Experience Track*, May 2005.

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

[22] Z. Gyöngyi and H. Garcia-Molina. Web spam taxonomy. In *First International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, Chiba, Japan, 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)*, pages 271–279, Toronto, Canada, Sept. 2004.

[24] M. R. Henzinger, R. Motwani, and C. Silverstein. Challenges in web search engines. *SIGIR Forum*, 37(2):11–22, Fall 2002.

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

[26] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the Web for emerging cyber-communities. *Computer Networks*, 31(11–16):1481–1493, 1999.

[27] R. Lempel and S. Moran. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. *Computer Networks*, 33(1–6):387–401, 2000.

[28] L. Li, Y. Shang, and W. Zhang. Improvement of HITS-based algorithms on web documents. In *The eleventh International Conference on World Wide Web Conference*, pages 527–535, Honolulu, Hawaii, USA, 2002. ACM Press.

[29] A. Mathes. Filler friday: Google bombing, Apr 2001. Online at http://uber.nu/2001/04/06/.

[30] Open Directory Project, 2005. http://dmoz.org/.

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

[32] P. Reddy and M. Kitsuregawa. Inferring web communities through relaxed cocitation and dense bipartite graphs. In *Proc. of Data Base Engineering Workshop*, 2001.

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

[34] M. Sahami, V. O. Mittal, S. Baluja, and H. A. Rowley. The happy searcher: Challenges in Web information retrieval. In *Trends in Artificial Intelligence: 8th Pacific Rim International Conference on Artificial Intelligence (PRICAI)*, volume 3157 of *Lecture Notes in Computer Science*, pages 3–12, Auckland, New Zealand, Aug. 2004. Springer.

[35] N. Shivakumar and H. Garcia-Molina. Finding near-replicas of documents on the web. In *WEBDB: International Workshop on the World Wide Web and Databases, WebDB.* LNCS, 1999.

[36] B. Wu and B. D. Davison. Identifying link farm spam pages. In *Proceedings of the 14th International World Wide Web Conference*, pages 820–829, Chiba, Japan, May 2005.

[37] B. Wu and B. D. Davison. Undue influence: Eliminating the impact of link plagiarism on web search rankings. In *Proceedings of the 21st Annual ACM Symposium on Applied Computing*, Dijon, France, Apr. 2006.

[38] M. J. Zaki and M. Ogihara. Theoretical foundations of association rules. In *In Proceedings of 3 rd SIGMOD'98 Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'98)*, Seattle, Washington, 1998.

[39] H. Zhang, A. Goel, R. Govindan, K. Mason, and B. V. Roy. Making eigenvector-based reputation systems robust to collusions. In *Proceedings of the Third Workshop on Algorithms and Models for the Web Graph*, Oct. 2004.