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

Baoning Wu and Brian D. Davison
Department of Computer Science & Engineering
Lehigh University
Bethlehem, PA 18015 USA
{baw4,davison}@cse.lehigh.edu

ABSTRACT

Link farm spam and replicated pages can greatly deteriorate link-based ranking algorithms such as 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 evaluations 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 [18].

Before the invention of link-based ranking algorithms, most search engine spam was related to content, and included techniques such as repeated key words and invisible text. With the advent of link-based ranking algorithms such as PageRank [6] and HITS [19] in the search engine world, such textual manipulation became less effective, and so the focus turned to link-based techniques.

When first introduced, algorithms like HITS (variants of which are found in search engines such as Teoma which powers Ask Jeeves) were able to triumph over 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 [5, 21, 22, 8].[1] For example, the top 10 authorities generated by HITS for query $weather$ are shown in Table 1.

[1] In manual review of HITS results over hundreds of queries in 2004 we found that more than 80 percent were distorted, containing recognizably irrelevant highly ranked authorities.

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

Table 1: Example of top list dominated by link farms.

There are three major factors that will degrade HITS results:

- “Mutually reinforcing relationships” [5] 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 [24, 16] 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 link target together with its anchor text as a basic unit (“complete hyperlinks”). 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 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 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.

The remainder of this paper is organized as follows. Section 2 reviews related work about link farms, community discovery, and duplicate Web pages. 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 idea of "mutually reinforcing relationships" is first introduced by Bharat and Henzinger [5]. The "TKC effect" is first mentioned by Lempel and Moran [21]. The authors propose the SALSA algorithm which is more resistant to the TKC effect than HITS. But SALSA is still vulnerable to a form of the TKC effect when many pages all point to each other [26].

Chakrabarti proposed using Document Object Models together with hyperlinks to beat nepotistic "clique attacks" [8]. Li et al. [22] found that HITS is vulnerable to the "small-in-large-out" situation. Gyongyi et al. describe a new algorithm, TrustRank, to combat Web spam [17]. Zhang et al. show how to use the "amplification factor" in PageRank results to detect link collusions [29]. Fetterly et al. use statistical analysis to find spam [13]. Amitay et al. [3] propose categorization algorithms to detect website functionality. They recognized that sites with similar roles exhibit similar structural patterns. Wu and Davison present a different method for finding link farm spam [28], 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. Finally, Davison [11] examined the viability of recognizing whether a hyperlink would be considered nepotistic.

Kumar et al. [20] used bipartite subgraphs and cores to find web communities. Flake et al. [14] proposed using maximum flow and minimum cut method to identify these commnuties. Reddy and Kitsuregawa also used bipartite graphs to find web communities [25]. Roberts and Rosenthal [26] propose simple algorithms to find clusters of web pages based on their outlink sets.

Several significant research papers about detecting duplicates of web pages by using shingles or chunks have been published [7, 27, 9, 4, 12].

3. Algorithm Details

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

1. Parse all pages for a query to get "complete hyperlinks" for each page and generate a document-hyperlink matrix. Filter out hyperlinks to pages within the same domain.
2. Find densely connected bipartite components within the matrix generated in step 1, by identifying the union of all satisfying complete bipartite graphs.
3. Change the adjacency matrix with weighted values according to the components found in step 2.
4. 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 [5]).
5. Calculate final ranking using the re-weighted link graph.

3.1 Complete hyperlinks

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.

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 1: Algorithm for finding bipartite components.

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.

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.

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 algorithm is shown in Figure 1. For a simple example, a starting matrix with 5 documents and 4 complete links is given in Table 2(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 Table 2(b).

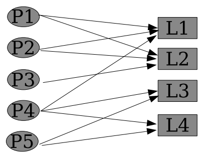
One problem with the above step is that sometimes an element will be marked incorrectly in the final matrix because it just has enough links within another bipartite component. As we can see from Table 2(b) that the $A[P4, L1]$ is still 1 because P1 and P2 have enough number of L1, but we don't want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

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 Table 2(b), we will get the matrix in Table 3(a) 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 3(a), P1 and P2 have two members, which is equal to the threshold 2. So the link P1-L1 and 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 $B[P4, P2]$.

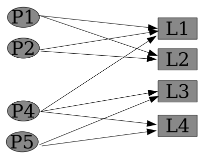
Remove all elements in matrix A that are not present in B. For

	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



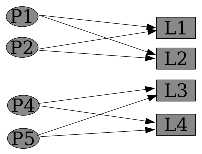
(a) Initial document-hyperlink matrix.

	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.

Table 2: Document-hyperlink matrix views.

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 2: Detail of the final adjusting step.

example, since P4-L1 is no longer present, that element will be set to 0. Then we get the matrix in Table 2(c). 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.

3.3 Down-weighting the adjacency matrix

As we mentioned before, there are two possibilities 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 [5, 26].

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

	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

(a) Document-document matrix B.

	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

(b) Down-weighted document-hyperlink matrix.

Table 3: Document-document and document-hyperlink matrices.

instead of 1, which is the default value. For example, in Table 2(c), $L1$ appears twice, so both $A[P1, L1]$ and $A[P2, L1]$ in Table 2(a) will be set to $1/2$. The final down-weighted matrix for this example is shown in Table 3(b).

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.

3.5 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 $O(nm*\text{min}(n,m))$. The final step has a complexity of $O(n^2m)$. So if $n = m$, in total the time complexity for detecting bipartite components is $O(n^3)$.

4. Experiments and Evaluation

In this section, we 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.

4.1 Data set

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.

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/

(b) 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/

(c) 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/

(d) CL-POP results.

Table 4: Results for query *rental car*.

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 used as the base data set for this query.

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 [19, 26, 8, 10, 21, 22], the names of the categories from the dmoz Open Directory Project [2], and popular queries from Lycos and Google [1, 15].

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. We show experimental results for two queries here.

Query: rental car. The top results for the four algorithms are in Tables 4(a)-4(d). 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.

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 listed in Table 5(c). HITS is dominated by pages from `http://www.teleactivities.com/`, where a lot of nepotistic links [20, 11] 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*. While CL-POP needs significantly less computation, it performs well, generating the same top list as 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 [20, 25]. There are some difficulties with this process, such as which duplicate to keep and which one to delete. 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 detection under the shingle test [7] while in fact many links within these two pages are common while much textual content is 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.

As mentioned earlier, 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 6(a). Although most of the top URLs are related to maps, several more famous websites for maps don't show up here, such as `maps.yahoo.com`. The reason for this is that there are many duplicate pages that replicate

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*.

`http://dmoz.org/Shopping/Publications/Maps/`, but a link to `maps.yahoo.com` is not included 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 6(b). We can see that 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.

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 6: Results for query *maps*.

The study included 28 participants, recording a total of 1688 preferences. The distribution of preference scores is shown in Table 7, in which 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". This is primarily a result of many fewer spam results by using CL-HITS and CL-POP.

We can also use this data to calculate precision at rank 10. If we allow both "quite relevant" and "relevant" choices to count, HITS achieved a precision at 10 of 23.6%, BH-HITS 42.8%, CL-HITS 77.2%, and CL-POP 72.5%.

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.

	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 7: Evaluation results.

One might wonder why we have used complete links, rather than something simpler, such as simply duplication of link targets. More importantly, a spammer who knows that our approach has been implemented could easily change the anchor text used, thwarting our recognition efforts. We consider both issues next.

We examined the effects of less-stringent link matching. Instead of using the exact match of the anchor text, for each anchor text we removed stop words and stemmed each word and put them in alphabetical order. This "bag of words" text replaced the original anchor text, and based on our user studies, this approached achieved a precision at 10 of 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.

When we compared the performance of using the links alone with the performance of using complete links on the same data set as above, the "quite relevant" and "relevant" categories summed to 66.4%, which is noticeably lower than the results of using complete links, suggesting that while using links alone may match more targets, it may be too aggressive, causing too many good links to be down-weighted.

Finally, we argue that our approach (especially the bag of words version) is expected to be an effective deterrent when its use is public knowledge. Most link farm spam is successful today not only because of an increase in perceived quality, but also as a result of the effects of link bombing [23]. With the knowledge of our algorithm, 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 markedly reduced by the use of different anchors).

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. 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.

7. References

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

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

[3] E. Amitay, D. Carmel, A. Darlow, R. Lempel, and A. Soffer. The connectivity sonar: Detecting site functionality by structural patterns. In *14th ACM Conf. on Hypertext and Hypermedia*, pages 38–47, 2003.

[4] 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.

[5] K. Bharat and M. R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proc. of 19th SIGIR*, pages 104–111, Melbourne, AU, 1998.

[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, 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.

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

[9] J. Cho, N. Shivakumar, and H. Garcia-Molina. Finding replicated Web collections. In *Proc. 2000 SIGMOD*, pages 355–366, 2000.

[10] D. Cohn and H. Chang. Learning to probabilistically identify authoritative documents. In *Proc. 17th ICML Conf.*, 2000.

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

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

[13] 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.

[14] G. Flake, S. Lawrence, and C. L. Giles. Efficient identification of web communities. In *Sixth ACM SIGKDD Conference*, pages 150–160, Boston, MA, August 2000.

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

[16] Z. Gyöngyi and H. Garcia-Molina. Web spam taxonomy. In *1st AIRWeb Workshop*, 2005.

[17] Z. Gyongyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with TrustRank. In *Proceedings of the 30th VLDB Conference*, Sept. 2004.

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

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

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

[21] 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.

[22] L. Li, Y. Shang, and W. Zhang. Improvement of HITS-based algorithms on web documents. In *Proc.of 11th International WWW Conference*, pages 527–535. ACM Press, 2002.

[23] A. Mathes. Filler Friday: Google Bombing, Apr 2001. Online at http://www.uber.nu/2001/04/06/.

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

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

[26] 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.

[27] N. Shivakumar and H. Garcia-Molina. Finding near-replicas of documents on the web. In *Proc. of WebDB workshop*. LNCS, 1999.

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

[29] 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.