A Spamicity Approach to Web Spam Detection[*]

Bin Zhou
Simon Fraser University
bzhou@cs.sfu.ca

Jian Pei
Simon Fraser University
jpei@cs.sfu.ca

Zhaohui Tang
Microsoft AdLab
zhaotang@microsoft.com

Abstract

Web spam, which refers to any deliberate actions bringing to selected web pages an unjustifiable favorable relevance or importance, is one of the major obstacles for high quality information retrieval on the web. Most of the existing web spam detection methods are supervised that require a large and representative training set of web pages. Moreover, they often assume some global information such as a large web graph and snapshots of a large collection of web pages. However, in many situations such assumptions may not hold.

In this paper, we study the problem of unsupervised web spam detection. We introduce the notion of spamicity to measure how likely a page is spam. Spamicity is a more flexible and user-controllable measure than the traditional supervised classification methods. We propose efficient online link spam and term spam detection methods using spamicity. Our methods do not need training and are cost effective. A real data set is used to evaluate the effectiveness and the efficiency of our methods.

1 Introduction

Ranking web pages is essential in web search and search engines. Due to a huge number of business opportunities brought by popular web pages, many tricks have been attempted to affect the search results and the rankings in search engines. Conceptually, *web spam* refers to any deliberate actions that bring to selected web pages an unjustifiable favorable relevance or importance. Web spam seriously hurts the quality of information retrieval on the web. Combating web spam has become more and more important in web search.

Detecting web spam is a challenging web mining task. The first generation of web spam detection methods often adopt a supervised learning model (a brief review is in Section 2). A training set consisting of pages labeled spam or normal is used to train a classifier, and the classifier is used to categorize other web pages.

However, there is some subtlety about modeling web spam detection as a traditional classification problem. The critical difference between a popular web page and a spam page is whether the popularity is justifiable. However, the measurement of justifiability is often subtle and subjective. In other words, it is often hard to label a web page absolutely spam or non-spam. Obtaining a reliable training data set for effective web spam detection is difficult, if possible at all.

Moreover, the scale of the web is huge, and keeps increasing with a fastened pace. It is hard to collect and maintain a training set sufficient for confident spam detection. New tricks emerge from time to time. New training samples have to be captured in order to combat new tricks.

In addition, the applicability of spam detection methods is an important concern. Most of the existing methods assume explicitly or implicitly that the spam detection methods are run at search engine sites where the detectors have good knowledge about the web including the web graph and the snapshots of the web. However, this assumption may not always hold. Spam detection is often required in practice, too, for off-search engine applications. By off-search engine applications, we refer to the scenarios where spam detection has to be conducted on sites where both the knowledge about the web and the computational resources are limited. Particularly, such a spam detector does not have the global knowledge about the web such as the web graph and the snapshots of many web pages. Moreover, the computational resources are constrained so that constructing a large web graph or crawling many web pages are not an acceptable solution. This is also partly due to the online response requirement.

Intelligent web browsers on personal computers are an example of such off-search engine applications. A web page currently shown in an intelligent web browser may have multiple links pointing to other web pages. It is desirable that such an intelligent browser can online detect whether those pages are spam or not. The

[*]We thank the anonymous reviewers for their insightful comments which help to improve the quality of this paper. This research is supported in part by an NSERC Discovery Grant and a grant from Microsoft AdLab. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

detection can annotate the current web page and give users more information and more control in the courses of browsing. In such a case, the spam detection has to be conducted on the browser site which may not have the information about the web graph and the snapshots of the web. Moreover, the spam detection has to be online since a user may not want to wait long for a detection result. To the best of our knowledge, all existing spam detection methods are based on large training data sets and/or global knowledge about the web. They may not be applicable to off-search-engine applications.

In this paper, we make the following progress. First, we propose to use spamicity to measure how likely a web page is spam. Spamicity is a more flexible and user-controllable measure than the traditional classification methods. Second, we propose efficient link spam and term spam detection methods based on spamicity. Our spamicity-based methods can return the spamicity score of a web page without any threshold. The score gives the degree of spamicity of the web page. This procedure does not need any training and is cost effective. Last, we use a real data set to evaluate the effectiveness and the efficiency of our spam detection methods using spamicity. For comparison, we try to find a threshold fitting the domain experts' judgment on spamicity. However, the threshold may be different for various users. A user can tune the threshold to reflect her/his tolerance of web spam.

The rest of the paper is organized as follows. In Section 2, we review related work. Link spam detection and term spam detection are addressed in Sections 3 and 4, respectively. A systematic empirical evaluation using a real data set is reported in Section 5. The paper is concluded in Section 6.

2 Related Work

In this section, we review some recent web spam detection methods very briefly. Limited by space, a systematic survey is far beyond the scope of the paper.

Most of the popular web search engines currently adopt some link-based ranking algorithms, such as PageRank [12] and HITS [10]. Driven by the huge potential benefit of promoting rankings of pages, many attempts have been conducted to boost page rankings by making up some linkage structures, which are known as link spam [8]. Because PageRank scores are determined based on the link structures of the web, PageRank is a natural target of link spam. Gyöngyi et al. [8] referred link spam to the cases where spammers set up structures of interconnected pages, called link spam farms, with the only purpose to boost the link-based ranking.

Some methods have been proposed to detect link spam. For example, Fetterly et al. [5] used statistical analysis to detect link spam. Features like in-degrees and out-degrees are used to conduct the analysis, and outliers are marked as spam candidates. Wu and Davison [14] proposed a supervised learning algorithm which starts with a seed set of link spam pages, and then identifies other link spam pages using links to and from the seed set. Recently, Gyöngyi et al. [7] introduced the concept of spam mass, a measure of the impact of link spam on a page's ranking. They discussed how to estimate spam mass and how the estimations can help to identify pages that benefit significantly from link spam.

Some other link spam detection methods resemble PageRank computation. Benczur et al. [2] proposed a method called SpamRank, which is based on the concept of personalized PageRank that detects pages with an undeserved high PageRank score. Gyöngyi et al. [9] developed an algorithm, TrustRank, to combat link spam. The basic assumption of TrustRank is that good pages usually point to good pages and seldom have links to link spam pages. Their method first selects a set of labeled good seed pages and assigns high trust scores to them. Then, it follows an approach similar to PageRank: the trust scores are propagated via out-links to other web pages. Finally, after convergence, the pages with high trust scores are categorized as good pages. However, TrustRank is vulnerable if the seed set used by TrustRank is not sufficiently representative to cover different topics on the web. Also, for a given seed set, TrustRank has a bias towards larger communities. To address the above issues, Wu et al. [15] proposed the use of topical information to partition the seed set and calculate the trust scores for each topic separately. A combination of these trust scores for a page is used to determine its ranking.

Term spam is the other type of web spam which disguises the content of a page so that it appears relevant to popular searches. Most of the term spam detection methods proposed so far adopted statistical analysis.

For example, in [5], Fetterly et al. studied the prevalence of term spam based on certain content-based properties of web sites. They found that some features, such as long host names, host names containing many dashes, dots and digits, as well as little variation in the number of words in pages within a site are good indicators of term spam pages. Later, in [6], Fetterly et al. investigated the special case of "cut-and-paste" content spam, where web pages are mosaics of textual chunks copied from legitimate pages on the web. They presented methods to detect such pages by identifying popular shingles. Recently, Ntoulas et al. [11] presented a number of heuristic methods for detecting term spam that essentially extend the previous work [5, 6]. Some of those methods are more effective than the others, however, the methods may not identify all kinds of term spam pages when used in isolation. Thus, [11] combined the term spam detection methods to create a C4.5 classifier to detect term spam pages.

All the existing methods assume explicitly or implicitly that the detector has some "global" knowledge about the web, such as the web graph and an extensive collection of web page snapshots so that statistics can be derived. Moreover, most of the existing methods require a training data set containing a good number of representative spam pages and/or normal pages. As analyzed in Section 1, those assumptions may not hold all the time.

3 Link Spamicity and Link Spam Detection

In this section, we first present a brief introduction to link spam. Then, we propose a utility-based link spamicity. Last, we discuss how the link spamicity of a web page can be figured out efficiently using a small number of "link search" queries to search engines.

3.1 Link Spam Web links can be modeled as a directed web graph $G = (V, E)$, where V is the set of web pages, and E is the set of hyperlinks. A link from page p to page q is denoted by an edge $p \rightarrow q$. An edge $p \rightarrow q$ can also be written as a tuple (p, q). A page p may have multiple hyperlinks pointing to page q, however, in the web graph, only one edge $p \rightarrow q$ is formed. Hereafter, by default our discussion is about a directed web graph $G = (V, E)$.

The link-based ranking methods such as PageRank and HITS are popularly used by major search engines in ranking web pages. PageRank measures the importance of a page p by considering how collectively other web pages point to p directly or indirectly. Formally, for a web page p, the *PageRank score* is defined as

$$PR(p, G) = d \sum_{p_i \in M(p)} \frac{PR(p_i, G)}{OutDeg(p_i)} + \frac{1-d}{N},$$

where $M(p) = \{q | q \rightarrow p \in E\}$ is the set of pages having a hyperlink pointing to p, $OutDeg(p_i)$ is the out-degree of p_i (i.e., the number of hyperlinks from p_i pointing to some pages other than p_i), d is a *damping factor* which models the random transitions on the web, and $N = |V|$ is the total number of pages in the web graph. The second additive term on the right hand side, $\frac{1-d}{N}$, is traditionally referred to as the random jump probability and corresponds to a minimal amount of PageRank score that every page gets by default.

Link spam is to deliberately build auxiliary pages and links to boost the PageRank or other link-based ranking score of the target page. Due to the extensive adoptions of the link-based ranking metrics, link spam has been used by many spam pages. Basically, link spammers build some artificially well constructed link structures with the only purpose to boost the ranking scores of the target pages. In most of the cases, such structures are referred to as *link spam farms*. A single target link spam farm consists of three parts: a single target page whose ranking is boosted, a reasonable number of boosting pages that deliberately improve the ranking of the target page, and some external links accumulated from pages outside the link spam farm, such as public blogs and forums.

3.2 Utility-Based Link Spamicity Link spam is typically a local activity. In order to boost the ranking of one page, a small number of pages and links (comparing to the whole web) are created deliberately. Driven by this critical insight, in order to determine whether one target page is link spam or not, we can extract some important neighbor pages to capture the "local environment" of the target page. We can calculate the utility-based link spamicity score, that is, how the PageRank score of the target page is boosted, to evaluate the likelihood that the target page is link spam.

Simply selecting those k-neighbor pages is not a precise way to capture the local environment of the target page, since different neighbor pages may have different importance to the ranking score of the target page. How can we capture the important neighbor pages and the local environment of a target page?

Consider a web graph $G = (V, E)$. For a subset of vertices $H \subset V$, the *induced subgraph* on H with respect to PageRank calculation is $G(H) = (V, E')$ where $E' = E - \{p \rightarrow q | p \notin H\}$. Please note that only the out-links of vertices not in H are removed, since removing the vertices in $(V - H)$ may affect the outdegrees of some vertices in H, and further affect the PageRank score calculation. Moreover, the distance from page p to page q is the number of edges in the shortest directed path from p to q.

A *page farm model* was proposed in [17]. For given parameters θ $(0 < \theta < 1)$ and integer $k > 0$, a *(θ, k)-farm* is a minimal set of pages whose distances to p are no more than k, and whose induced subgraph contributes to a θ portion of the PageRank score of a target page. The page farm of target page p as an induced subgraph is denoted by $Farm(p)$. As shown in [17, 16] using real data sets, when $\theta \geq 0.8$ and $k \geq 3$, the (θ, k) farms capture the local environments of web pages accurately.

Using page farms, we can develop a utility-based link spam detection method. Intuitively, if p is link spam, then $Farm(p)$ should try to achieve the PageRank of p as high as possible. We can calculate the maximum PageRank score using the same number of pages and the same number of links as $Farm(p)$ has. The utility of the page farm of p is the ratio of the PageRank of p against the maximum PageRank that can be achieved. The utility can be used as a measure on the likelihood that p is link spam. If the utility is closer to 1, the page is more likely to be link spam.

What is the largest PageRank score that a farm of n pages and l links can achieve?

THEOREM 3.1. (MAXIMUM PAGERANK SCORES) *Suppose that $Farm(p)$ contains n pages $p_1, \ldots, p_n$ except for p, and l hyperlinks $e_1, \ldots, e_l$. The following structure maximizes the PageRank score of p.*

$$e_i = \begin{cases} p_i \to p & (1 \le i \le n) \\ p \to p_{i-n} & (n+1 \le i \le 2n) \\ p_{\lceil \frac{i-2n}{n-1} \rceil} \to p_{h(i)} & (2n+1 \le i \le l) \end{cases}$$

where $h(i) = 1 + (i - 2n - \lceil \frac{i-2n}{n-1} \rceil (n-2) + 1) \bmod n$.

Proof sketch. The structures are shown in Figure 1, where the third situation in the theorem is elaborated in Figures 1(c) and (d), (c) being a special case of (d). The proof can be constructed by an induction on the number of edges in the farm. Limited by space, we omit the details here. ■

Based on Theorem 3.1, we denote by $PR_{max}(n, l)$ the maximum PageRank score of the target page that a page farm of n pages and l links can achieve. A page farm of n pages and l hyperlinks is called an *optimal spam farm* if it achieves the maximum PageRank score of the target page.

DEFINITION 1. (UTILITY-BASED LINK SPAMICITY) For a target page p, we define the *utility-based link spamicity* of p as $ULSpam(p) = \dfrac{PR(p)}{PR_{max}(|V|, |E|)}$. ■

The utility-based link spamicity of a web page is between 0 and 1. The higher the utility-based link spamicity, the more the page farm is utilized to boost the PageRank of the target page. Spammers (i.e., the builders of spam web pages) build up the "link spam farms" with the only purpose to boost the rankings of the target pages as much as possible. The optimal spam farms do not commonly happen on the web, because they are quite different from those normal page farms.

Moreover, since optimal spam farms are highly regular as indicated by Theorem 3.1, a search engine may easily detect the optimal spam farms. To disguise, a spammer may modify the optimal spam farm. However, the utility of the page farm of a spam page has to be high in order to obtain a high PageRank score using a relative small number of supporting pages. Using the utility-based link spamicity, we can still capture the disguised link spam.

3.3 Efficient Link Spam Detection In this section, we present an online link spam detection method based on the utility-based link spamicity. To make our method extensively applicable, we assume that the whole web graph is unavailable. Consequently, we have to obtain the local link structure of a page by parsing the content of the page and querying web search engines. The major search engines including Google, Yahoo, and Microsoft Live Search can answer "link search" queries. A user can submit one URL to a search engine. The engine returns a list of pages having a hyperlink pointing to the submitted URL. Moreover, given a page p, we can parse the content of the page so as to know the pages that p points to.

The major cost in link spam detection can be divided into two parts. First, the *search engine querying load (in-link search)* is the cost of sending a link search query to a search engine and obtaining the list of pages that have a hyperlink pointing to the query page. The average cost of an in-link search is denoted by c_1. Second, the *web page out-link parsing load (out-link search)* is the cost of retrieving a page and extracting the outgoing links in the page. The average cost of an out-link search is denoted by c_2.

3.3.1 Page Contribution and Path Contribution Utility-based link spamicity is based on page farms. A major operation to find page farms is to calculate the PageRank score of a page in various induced subgraphs. The notions of page contribution and path contribution can help.

Consider two pages v and p in a set of pages V. What is the contribution of v to the PageRank of p? If $v = p$, according to the definition of PageRank score, $\frac{1-d}{N}$ is the amount of PageRank score that every page gets by default, and thus can be regarded as the self-contribution. If $v \neq p$, intuitively, the contribution from v to p is the decrease of the PageRank score of page p after we void page v.

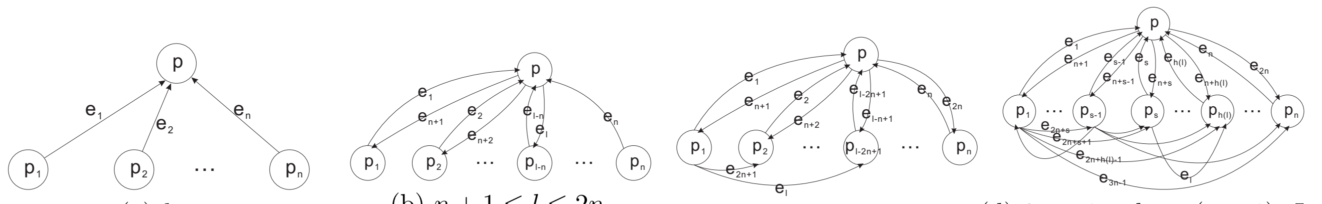
DEFINITION 2. (PAGE CONTRIBUTION) For a target page $p \in V$, the *page contribution* of page $v \in V$ to the PageRank score of p is

$$PCont(v,p) = \begin{cases} PR(p, G) - PR(p, G(V - \{v\})) & (v \neq p) \\ \frac{1-d}{N} & (v = p) \end{cases}$$

where d is the damping factor, and N is the total number of pages in the web graph. ■

Computing contributions page by page is costly. Can we reduce the cost effectively? One idea is to compute contributions path by path.

DEFINITION 3. (PATH CONTRIBUTION) Consider web graph $G = (V, E)$ and target page $p \in V$. Let $P =$



(a) l=n. (b) $n+1 \leq l \leq 2n$. (c) $2n+1 \leq l \leq 3n-1$. (d) $2n+1 \leq l \leq n(n+1)$. In the figure, $s = \lceil \frac{l-2n}{n-1} \rceil$.

Figure 1: Achieving the maximum PageRank scores.

$v_0 \rightarrow v_1 \rightarrow \cdots \rightarrow v_n \rightarrow p$ be a directed path from v_0 to p in the graph. The *path contribution* to the PageRank of p from P is defined as

$$LCont(P,p) = \frac{1}{N} d^{n+1}(1-d) \prod_{i=0}^{n} \frac{1}{OutDeg(v_i)},$$

where $OutDeg(v_i)$ is the out-degree of page v_i, and N is the total number of pages in the web graph. ■

As shown in [3], PageRank scores can be calculated using the path contributions.

$$PR(p,G) = \frac{1-d}{N} + \sum_{v \in W(p)} \Big(\sum_{P \in DP(v,p)} LCont(P,p) \Big),$$

where $W(p) = \{v | \text{there is a directed path from } v \text{ to } p\}$, $DP(v,p) = \{\text{directed path } P \text{ from } v \text{ to } p\}$, and N is the total number of pages in the web graph.

Moreover, page contribution can also be calculated using path contributions.

Corollary 3.1. (Page and path contributions) *For vertices p and q in web graph $G = (V,E)$, if the in-degree of q is 0, i.e., $InDeg(q) = 0$, then $PCont(q,p) = \sum_{\text{path } P \text{ from } q \text{ to } p} LCont(P,p)$. If $InDeg(q) > 0$, then*

$$PCont(q,p) = \sum_{\text{path } P_1 \text{ from } q \text{ to } p} LCont(P_1,p) + \sum_{v \in W_q(p)} \sum_{\text{path } P_2 \text{ from } v \text{ to } p \text{ through } q} LCont(P_2,p),$$

where $W_q(p) = \{v | \text{there is a directed path from } v \text{ to } p \text{ through } q\}$. ■

3.3.2 Online Utility-Based Link Spam Detection

The problem of utility-based link spam detection can be defined as follows. For a target page p and a utility threshold α, determine whether the utility-based link spamicity of p is greater than or equal to α. If so, then p is a suspect of link spam and the spamicity of p should be reported. Otherwise, p is reported to be not-spam.

Based on page contribution and path contribution, we can have the following greedy algorithm to extract approximate (θ,k)-farms. For a given target page p, we greedily add pages with the highest page contribution to p into the farm until p achieves a θ portion of the global PageRank score of p. Using the page farm, we can calculate the utility-based link spamicity score. In the later analysis, we call this algorithm the *local greedy search* method.

In this local greedy search method, we have to extract the whole (θ,k)-farm of the target page. We have to conduct an in-link search and an out-link search for every page in the farm. To reduce the search cost, can we detect link spam but avoid extracting the whole page farm in many cases?

A critical observation is that, if pages are added in the page contribution descending order, the utility of adding new pages to improve the PageRank of the target page decreases monotonically.

Theorem 3.2. (Monotonicity) *Let $G = (V,E)$ be the web graph. For a target page p, let q_1 be a page which has a hyperlink to p and the PageRank score of p in $G(\{p,q_1\})$ is maximized. Let q_i $(i > 1)$ be a page which has a hyperlink to at least one of $p, q_1, \ldots, q_{i-1}$ and the PageRank score of p in $G(\{p,q_1,\ldots,q_{i-1},q_i\})$ is maximized. Let $PR_j(p)$ $(j \geq 1)$ be the PageRank of p in $G(\{p,q_1,\ldots,q_j\})$ and N_i be the number of edges in $G(\{p,q_1,\ldots,q_i\})$. Then,*

$$\frac{PR_i(p)}{PR_{max}(i,N_i)} \geq \frac{PR_{i+1}(p)}{PR_{max}(i+1,N_{i+1})}.$$

Proof sketch. The proof can be given by an induction on the distance (i.e., the number of edges) from q_i to p. The intuition is that when the distance increases (i.e., more remote neighbors are added into the page farm), the utility of the new page towards the improvement of the PageRank score of p decreases, since a page in the optimal structure always links to the target page directly. Limited by space, we omit the details here. ■

Theorem 3.2 leads to an efficient utility-based link spam detection method. We initialize the page farm of

p by p itself. In each iteration, let F be the current farm. Among all pages that are connected to F, we greedily add a page to the current farm which makes the largest improvement on the PageRank of p in the updated farm. The iteration continues until the utility-based link spamicity computed using the current farm is lower than the utility threshold, or all the pages whose distance to p is up to k are in the farm. Theorem 3.2 guarantees that once the utility-based spamicity is lower than the threshold in an iteration, it will never come back and thus the page is not spam. On the other hand, if all pages whose distance to p is at most k are searched, the page is suspect of link spam and should be output. In the later analysis, we call this algorithm the *monotone greedy search* method.

The advantage of the pruning using Theorem 3.2 is that, for many non-spam pages, the utility drops quickly as more pages are added to the farm. As shown in our experimental results, we often only need to conduct a small number of in-link and out-link searches to determine that a page is not spam. We do not need to extract the whole page farm or a large portion of it. Another advantage is that we even do not need the value of θ.

Only when a page is link spam, we have to extract all neighbor pages up to distance of k. Nevertheless, most of the pages on the web are not spam. Some recent studies [5, 8] indicated that about 10% to 15% of the pages are spam.

4 Term Spamicity and Term Spam Detection

In this section, we tackle the problem of online term spam detection. We first give a brief introduction to term spam. Then, we propose two methods: a utility-based method and a characteristics-based method.

4.1 Term Spam The term-based ranking methods such as TFIDF [1] adopted by web search engines are victims of term spam. The previous studies (e.g., [8]) show that the algorithms used by search engines to rank web pages based on their content information use various forms of the fundamental TFIDF metric. A web page p and a search query Q can be regarded as a set of keywords. The *TFIDF score* of a web page p with respect to a search query Q is defined as

$$TFIDF(p, Q) = \sum_{t \in p \cap Q} TF(t) \times IDF(t),$$

where $TF(t)$ is the term frequency of keyword t in p, and $IDF(t)$ is the inverse document frequency of keyword t which is the total number of documents in the collection divided by the number of documents that contain t.

A web page p can be divided into several parts, such as the page body, the page title, the meta tags in the HTML header, the URL address, and the anchor fields. Each part is called a *field*. Except for anchor fields which contain anchor text associated with URLs that point to p in pages other than p, all other fields are in p.

The appearances of keywords in different fields carry different significance. For example, a keyword appears in the page title may have higher importance than that in the page body. When ranking web pages, in order to evaluate the content relevance, search engines may assign different weights to different fields. For example, the keywords in the text fields are used to determine the global relevance of the page with respect to a specific query.

Term spam refers to tricks that tailor the contents of text fields to make spam pages relevant for some queries. Spammers want to increase the TFIDF scores of term spam pages as much as possible. Since the IDF score for a specific keyword is often hard to be affected substantially by a spammer, the primary way to increase the TFIDF score is to increase the frequencies of keywords within some specific text fields of the term spam pages.

According to the TFIDF-based ranking methods, spammers may have two different strategies to deliberately influence the ranking results. On the one hand, a spammer can make a term spam page relevant to a large number of search queries. In other words, the term spam page receives a positive TFIDF score for a large set of queries. This can be done by including a large number of distinct keywords in the page. On the other hand, a spammer can make a term spam page highly relevant to a specific search query. That is, the page receives a high TFIDF score for the given query. This can be done by repeating some targeted keywords in the page.

4.2 Utility-Based Term Spamicity For a page, we can calculate a term spamicity score to measure the likelihood of the page being term spam. In order to determine whether a page p is term spam, we only need to parse page p and those pages having a link pointing to p (to collect the anchor fields). As the first try, we propose the utility-based term spamicity here.

If page p is term spam, to be relevant to a search query Q, p should try to achieve the TFIDF score as high as possible. We can calculate the maximum TFIDF score for a query Q using the same number of keywords that p has. The utility of page p with respect to term spam is the ratio of the TFIDF score of p against the maximum TFIDF score of p that can be achieved. The utility can be used as a measure of the likelihood that p is term spam. Intuitively, if the utility is closer to 1, the page is more likely to be term spam.

Then, what is the largest TFIDF score that a page

p with n keywords of l occurrences can achieve?

The TFIDF score of page p with respect to a query Q depends on the number of common keywords shared by p and Q. Spammers would like to increase the frequency of some keywords in their term spam pages. Once a query contains those keywords, the term spam pages are ranked high by search engines. To detect whether a web page is term spam, we can consider the keywords in page p as the targeted keywords to which the builder of the page wants to make p relevant. Generally, each keyword may have a different probability to appear in queries. Although the frequencies of keywords in queries can be obtained by search engines from their query logs, the information is generally not available for spammers and off search engine spam detectors. Here, we assume that each keyword takes the same possibility to appear in a query.

For a web page p of l keyword occurrences, let $Q = \{w_1, \cdots, w_n\}$ be the set of keywords in the page, and d_i $(i = 1, \ldots, n)$ be the inverted document frequency of word w_i. As perfect term spam, page p should get TFIDF scores as high as possible with respect to queries having keywords in Q. Therefore, we define the *term spam utility maximization problem* as to assign term frequency f_i to keyword w_i in p so that $\sum_{i=1}^{n} TFIDF(p, w_i)$ is maximized. We have the following result.

Theorem 4.1. (Maximum TFIDF Scores) *In the term spam utility maximization problem, let w_{i_0} be the keyword having the largest inverted document frequency. Then, assigning $f_{i_0} = 1 - \frac{n-1}{l}$ and $f_j = \frac{1}{l}$ $(1 \leq j \leq n, j \neq i_0)$ maximizes $\sum_{i=1}^{n} TFIDF(p, w_i)$.* ∎

Please note that the inverted document frequency of a keyword can be obtained by querying search engines using the keyword. Search engines return the approximation of the total number of documents containing the keyword. Based on Theorem 4.1, we denote by $TFIDF_{max}(p)$ the maximum TFIDF score that a page p can achieve.

Definition 4. (Utility-based term spamicity) For a target page p, the *utility-based term spamicity* of p is

$$UTSpam(p) = \frac{TFIDF(p, Q)}{TFIDF_{max}(p)},$$

where Q is the set of keywords in p. ∎

The utility-based term spamicity of a web page p is between 0 and 1. The higher the utility-based term spamicity, the more page p is utilized to boost the TFIDF score.

The assumption in the utility-based term spamicity that a spammer wants to maximize $\sum_{i=1}^{n} TFIDF(p, w_i)$ may be problematic. In general, different queries may carry different weights to spammers, since different keywords may have different probabilities to appear in queries. Next, we explore effective term spam detection by analyzing fields in a web page.

4.3 Characteristics-Based Spamicity

Comparing to link spam, term spam is even more likely a local activity. The possible fields where term spam exists are in the target page and those pages having a link directly pointing to the target page. For a target page p, we can extract the content of p and its 1-in-neighbor pages (i.e., those pages having a link pointing to p). We can examine the characteristics of the content information to evaluate the likelihood of term spam for the target page.

Fetterly et al. [5] and Ntoulas et al. [11] proposed some content-based heuristics to combat term spam. Beyond their work, we identify three other useful heuristics here to measure the likelihood of term spam for a web page.

4.3.1 Keyword Stuffing Detection

"Keyword stuffing" [8] is one popular way to create term spam web pages. The content of a web page is augmented with a number of popular keywords so that the page is relevant to some popular search queries. In order to raise the TFIDF score, spammers have to increase the term frequency as much as possible. An effective way to spam the content of a page is to repeat some keywords.

Keyword stuffing may happen to the fields of page body, page titles, page meta tags, and page anchor text. Here, meta tags are the information in the "HEAD" area of the HTML code of web pages. Information in this area is not seen by users of browsers. Instead, the information in meta tags are mostly used to communicate with web browsers and search engines such as declaring keywords or providing a content description. Anchor text is the visible text in a hyperlink of a page. Since the anchor texts associated with hyperlinks that point to a page p often describe very well the content of p, these anchor texts can be considered highly related to page p.

In order to detect keyword stuffing in the above fields, we define the following measures. The *keyword redundancy* $H_1(p)$ of a web page is the ratio of the total number of keywords in the body against the number of distinct keywords in the body. The *title keyword redundancy* $H_2(p)$ of a web page is the ratio of the number of keywords in the title against the number of unique keywords in the title. The *meta tag keyword redundancy* $H_3(p)$ in a web page as the ratio of the number of keywords in the meta tags against the number of unique keywords in the meta tags. The *anchor text keyword redundancy* $H_4(p)$ in a web page is the ratio of the number of keywords in the anchor texts against the number of unique keywords in the anchor texts.

HEURISTIC 1. (KEYWORD STUFFING DETECTION) *The larger the keyword redundancy, the more likely the page is term spam.* ■

4.3.2 Invisible Keywords in the Body The "keyword stuffing" tricks may easily be noticed by web users. Thus, a number of tricks have been exercised to "hide" term spam from users visiting spam pages. One simple way is called "content hiding" [8]. The spam keywords in a page can be made "invisible" to web users by simply setting the font in the same color as the background.

We define the *invisible rate* $H_5(p)$ of a web page as the ratio of the number of "invisible" keywords in the body against the total number of keywords in the body. A web page having a large invisible rate is likely term spam.

HEURISTIC 2. (INVISIBLE RATE) *The larger the invisible rate, the more likely the page is term spam.* ■

4.3.3 Page URL Spam Some previous studies [8] show that some search engines also break down the URL of a page into a set of keywords, which can be used to determine the ranking of the page. URL spam refers to the term spam tricks that some spam keywords are embedded in the URL address of the page.

Since search engines take into account the keywords in URL addresses, some spammers may want to create long URLs that include spam keywords.

We define the *URL keyword utility* $H_6(p)$ in a web page as the ratio of the total length of keywords in the URL address against the total length of the URL address. A web page having a large URL keyword utility is likely term spam.

HEURISTIC 3. (URL KEYWORD UTILITY) *The larger the URL keyword utility, the more likely the page is term spam.* ■

4.3.4 Characteristics-Based Term Spamicity The experimental results shown in Section 5 indicate that an individual heuristic may be not good enough to reflect the spamicity of a web page. To integrate all heuristics, we normalize the heuristic values (H_1 to H_6) into range $[0, 1]$, and then use the Minkowski distance to integrate all heuristics into a characteristics-based term spamicity measure. Minkowski distance is a general distance notion, and is widely used in various kinds of scenarios. Its distance order (γ in the definition) is tunable. In off-search engine applications, no training data can be used to assign weights to each heuristic. Intuitively, each heuristic focuses on one specific term spam activity, thus it is reasonable to make them carry the same weight. To keep our discussion simple, we still denote the normalized heuristic values as H_1 to H_6.

DEFINITION 5. (CHARACTERISTICS-BASED SPAMICITY) For a web page p, the *characteristics-based term spamicity* is

$$CTSpam(p) = \sqrt[\gamma]{\frac{\sum_{i=1}^{6} H_i(p)^\gamma}{6}}$$

where $\gamma > 0$ is the *Minkowski distance parameter* and $H_i(p)$ is the i-th term spam heuristic discussed before. ■

4.4 Efficient Term Spam Detection The problem of unsupervised term spam detection can be defined as, given a (utility-based or characteristics-based) term spamicity threshold and a web page, determine whether the term spamicity of the page is greater than or equal to the threshold. If so, then the page is a suspect of term spam. Otherwise, the page is not spam.

We can take a cost model similar to the one for link spam detection. Essentially, there are three major types of cost.

First, the *web page keyword parsing load (keyword search)* is to extract the keywords from a web page by retrieving the page and parsing it. We denote the average cost of parsing a web page by c_3. Second, the *search engine querying load (in-link search)* is needed to extract the 1-in-neighbor pages in order to extract the anchor texts. This cost is the same as c_1 in online link spam detection. Last, if the IDF scores of keywords are not available, then we also need to derive them by querying search engines. However, a detector can accumulate the information about IDF scores as more detections are conducted. We denote the average cost of a query to the search engine for a keyword by c_4.

Given a web page p and a term spamicity threshold β, we calculate its utility-based term spamicity score or characteristics-based term spamicity score. For the utility-based term spamicity, we only need to consider the content of the target page. Thus, the total cost for a target page p is $c_3 + n_1 c_4$, where n_1 is the total number of keywords whose IDF scores are not available at the detector side.

However, for the characteristics-based term spamicity, we not only consider the content of the target page, but also consider the content of pages that pointing to the target page. Thus, the total cost for a target page p is $c_3 + c_1 + n_2 c_1 = c_3 + (n_2 + 1)c_3$ where n_2 is the number of pages having a link pointing to p. Although the characteristics-based term spamicity needs more cost than the utility-based term spamicity, it is more accurate according to the experimental results.

5 Experimental Results

We use the webspam-UK2006 data set [4] recently released by the Search Engine Spam Project at Yahoo! Research Barcelona (http://aeserver.dis.uniroma1.it/webspam). The data set is the result of the effort of a team of volunteers. The *base data set* contains $77,862,535$ pages in the domain of .UK downloaded in May 2006 by the Laboratory of Web Algorithmics, Universitá degli Studi di Milano.

The *spam test collection* data set consists of $8,415$ different pages chosen from the base data set. A team of volunteers were asked to classify this set of pages as "normal", "spam" or "borderline". Moreover, the project organizers added two kinds of special votes: all the UK pages mentioned in the Open Directory Project (http://www.dmoz.org) in May 2006 are voted "normal", and all the UK pages ending in .ac.uk, .sch.uk, .gov.uk, .mod.uk, .nhs.uk or .police.uk are voted "normal". It is believed that the pages in those domains are rarely spam.

Whether a page is spam is labeled by assigning 1 point to each vote of "spam", 0.5 point to each vote of "borderline", and 0 point to each vote of "normal". The final label for a page is determined by the average of points from all votes on this page: an average of over 0.5 point is "spam", an average of less than 0.5 point is "normal", and an average of 0.5 point is "undecided".

Among the $8,415$ pages in the spam test collection data set, 176 pages are labeled as "borderline". However, the web spam detection methods proposed so far only classify web pages into two categories, "normal" pages or "spam" pages. In order to make a fair comparison and remove the noisy information, we discarded those pages labeled as "borderline" in the experimental evaluation. One thing we have to mention is that our spamicity-based spam detection method still can work well for those "borderline" pages, since the spamicity score of a page reveals the likelihood and the degree of one page being spam. Finally, $8,239$ pages in the data set that have a label either "normal" or "spam" were considered. Among the $8,239$ pages, 767 pages are labeled as "spam". By default, we use this data set in the empirical evaluation reported here.

To implement the term spam detection methods, we first extracted keywords from web pages. The Google stopwords (http://www.ranks.nl/tools/stopwords.html) were discarded. Then we counted the keyword frequencies. In order to estimate the IDF score of a keyword t, we submitted the keyword query t to a search

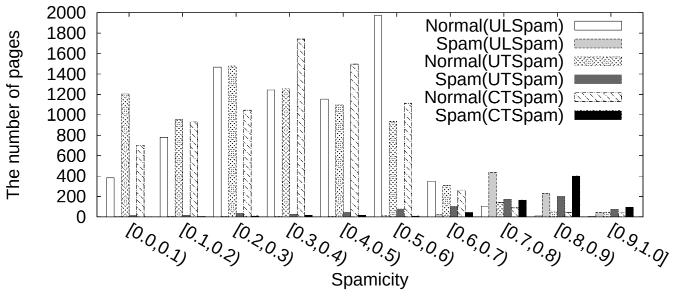


Figure 2: The effectiveness of spamicity measures.

engine and obtained the approximate number of pages containing t. The total number of indexed pages on the web was estimated by submitting one query "`*a*`" to a search engine. Finally we calculated the TFIDF scores and the maximum TFIDF scores for pages in the data set.

In the characteristics-based term spamicity, by default we set $\gamma = 2$. We observed that the effectiveness of the measure is not sensitive to γ.

All the experiments were conducted on a PC computer running the Microsoft Windows XP SP2 Professional Edition operating system, with a 3.0 GHz Pentium 4 CPU, 1.0 GB main memory, and a 160 GB hard disk. The program was implemented in C/C++ using Microsoft Visual Studio. NET 2003.

5.1 Effectiveness of Spamicity Measures We first test the effectiveness of the three spamicity measures proposed in this paper: utility-based link spamicity (ULSpam), utility-based term spamicity (UTSpam), and characteristics-based term spamicity (CTSpam). Figure 2 shows the distribution of normal and spam pages in groups with various ranges of spamicity scores. When the spamicity values are low, most pages are normal pages. When the spamicity values are high, most pages are spam pages. This clearly shows that the spamicity measures can discriminate normal pages and spam ones.

Characteristics-based term spamicity is an integration of 6 heuristic values. Figure 3 shows the effectiveness of the six individual heuristics. No single heuristic works perfectly. However, different heuristics are used to detect different spam tricks. The integrated effect turns out to be good. In fact, the CTSpam measure performs better than any single heuristic. This is consistent with the observations in [11].

In spam detection, we use a spamicity threshold to detect spam pages. Figure 4 shows the effect of spamicity thresholds on the accuracy, recall, and F-measure of the three spamicity measures. When the spamicity threshold increases, less pages are reported

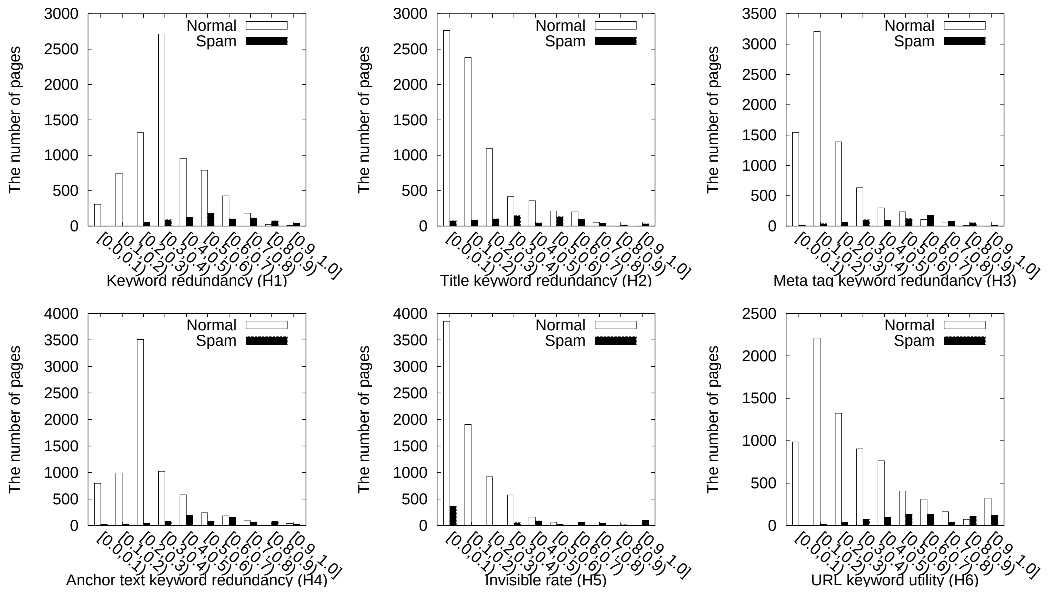


Figure 3: The effectiveness of the 6 heuristics for characteristics-based term spam detection.

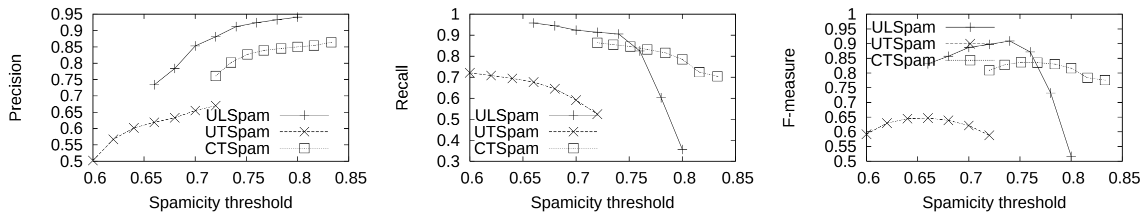


Figure 4: Comparison of the three spamicity measures in precision, recall, and F-measure.

as spam. The precision increases since those pages with high spamicity values are very likely spam. However, the recall decreases. Utility-based link spamicity has the best effectiveness. The F-measure curve shows that when the threshold is about 0.74, it has the best performance. Characteristics-based term spamicity performs better than utility-based term spamicity. As analyzed before, the assumption that each keyword has the same probability to appear in a query hurts the effectiveness of utility-based term spamicity. The characteristics-based term spamicity and the utility-based term spamicity perform the best when the thresholds are set to 0.76 and 0.65, respectively.

5.2 Efficiency of Spam Detection We evaluate the efficiency of our spam detection methods. We also compare our methods with some state-of-the-art methods.

5.2.1 Online Link Spam Detection The monotone greedy search is an approximation of the local greedy search for utility-based link spam detection. Figure 5 compares the effectiveness of the two methods as well as SpamRank [2], the only existing method that detects link spam by assigning a spamicity-like score and does not need supervised training. SpamRank assumes that link spam pages have a biased distribution of pages that contribute to the undeserved high PageRank score of the spam page. SpamRank penalizes pages that originate a suspicious PageRank share and personalizes PageRank on the penalties. We tried our best to implement the method as described in [2]. We normalized the SpamRank values into the range $[0, 1]$. SpamRank has to access the whole web graph, and conduct an iterative PageRank-like computation.

The spam detection quality of the monotone greedy search is very close to that of the local greedy search. The local greedy search method is about 5% better. The utility-based link spamicity method performs better than SpamRank. The results show that the utility-based link spamicity can capture the inherent features of link spam better.

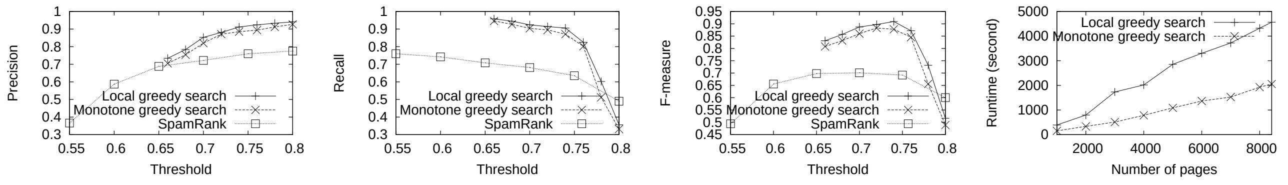


Figure 5: Comparison of local greedy search, monotone greedy search, and SpamRank in precision, recall, and F-measure, and runtime.

	All pages		Normal pages		Spam pages	
	$\#c_1$	$\#c_2$	$\#c_1$	$\#c_2$	$\#c_1$	$\#c_2$
Local	123	218	126	219	94	208
Monotone	57	103	51	90	114	225

Table 1: The average cost per page in the two utility-based link spam detection methods ($\#c_1$: number of in-link searches, $\#c_2$: number of out-link searches).

	$\#c_1$	$\#c_3$	$\#c_4$
Utility-based	0	1	131
Characteristics-based	1	23	/

Table 2: The average cost per page in the utility-based term spamicity and the characteristics-based term spamicity methods ($\#c_1$: the number of in-link searches, $\#c_3$: the number of pages parsed, $\#c_4$: the number of IDF scores computed).

In the figure, we also compare the runtime of the local greedy search and the monotone greedy search. Both methods have a roughly linear scalability. The monotone greedy search is more than 2 times faster. To understand the difference in runtime, we collect the average cost in both methods in Table 1. The monotone greedy search method conducts much less in-link and out-link searches on average than the local greedy search on all pages and on the normal pages. This explains the advantage of the monotone greedy search method. On spam pages, the monotone greedy search method has 21% more in-link searches, and 8% more out-link searches. However, as indicated by the ground truth, most of the pages on the web are normal ones.

We do not plot its runtime here, since it computes the spamicity of one single page only. When the whole web graph is available in main memory, the ratio of the total runtime of SpamRank over the total number of web pages in the graph is about 0.5 second, which is twice of average cost in the monotone greedy search method.

From the above experiments, we can see that the monotone greedy search method achieves a good trade-off between spam detection quality and efficiency. It also outperforms SpamRank in both detection quality and efficiency.

5.2.2 Online Term Spam Detection Figure 6 compares the runtime of the utility-based term spam detection method and the characteristics-based term spam detection method. To explain the difference in efficiency, Table 2 examines the average cost in both methods. As analyzed in Section 4.4, characteristics-based term spam detection is more costly. On the other hand, it is more accurate in detection quality.

We compare our spamicity-based term spam detection methods with the decision tree method by Ntoulas et al. [11]. In that method, 10 content-based heuristics are used to extract features of term spam pages and a decision tree is constructed using $C4.5$ [13].

We use the $C4.5$ executable available due to Quinlan. In order to evaluate the accuracy of the classifier, we employ 10-fold cross validation. The whole data set is divided randomly into 10 equally-sized partitions, and 10 training and testing steps are conducted. In each step, we use 9 partitions to train the classifier and the remaining 1 partition to test its effectiveness. 10 features including the total number of words in the page, the total number words in the title as proposed in [11] are used to train the decision tree.

The precision of the spam detection using the decision tree method is 82.0%, and the recall is 83.1%. The decision tree method outperforms the utility-based term spamicity method, but is not as good as the characteristics-based term spamicity method. This result indicates that the heuristics identified in this paper seem better in term spam detection.

5.3 Integrating Spam Detection Currently, the ranking metrics used by most popular web search engines are not purely link-based or purely term-based. For example, http://searchenginewatch.com indicated that more than 100 different factors are taken into account when Google ranks web pages. Consequently, spammers use both link spam and term spam tricks together most of the time to achieve the largest spam benefit. The web spam detection methods proposed so

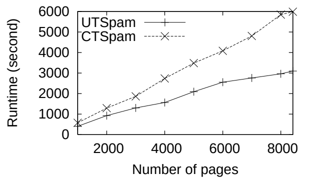


Figure 6: Term spam detection runtime.

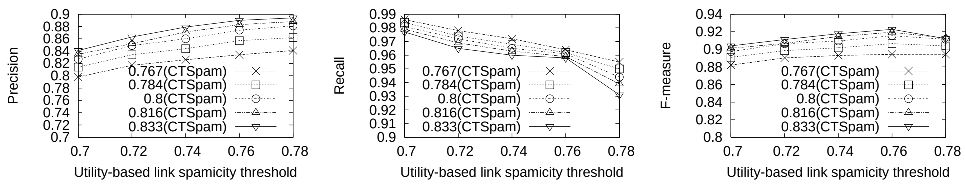


Figure 7: Integrating link spamicity and term spamicity in web spam detection.

far treat link spam detection and term spam detection separately. Thus, an effective way to combat web spam should combine the two categories of spam detection methods.

In our methods, the spamicity score of a page is used to evaluate the likelihood of a page is spam. A simple yet effective way to detect web spam pages is by setting two spamicity thresholds, one threshold α for link spamicity, and the other threshold β for term spamicity. For each page, we calculate the link spamicity score and term spamicity score, respectively. A web page is reported as "normal" if and only if both of its link spamicity and term spamicity are less than the corresponding spamicity thresholds.

Figure 7 shows the effectiveness of using characteristics-based term spamicity (CTSpam) and utility-based link spamicity (ULSpam) with respect to different spamicity thresholds. Combining the two methods can clearly improve the effectiveness of spam detection. When the utility-based link spamicity threshold is set to 0.76 and the characteristics-based term spamicity is set to 0.833, the integration method has the best performance.

Comparing Figure 7 to Figure 4, we can clearly see that the integration method is more robust with respect to thresholds. This indicates that the two methods are relatively complementary. They can collaborate to achieve good accuracy.

6 Conclusions

In this paper, we studied the problem of unsupervised web spam detection. We introduced the notion of spamicity to measure how likely a page is spam. Spamicity is a more flexible and user-controllable measure than the traditional classification methods. We proposed efficient link spam and term spam detection methods. Our methods do not need training and are cost effective. A real data set is used to evaluate the effectiveness and the efficiency of our methods. The experimental results clearly showed that our methods are effective and efficient to detect spam pages.

There is a lot of future work can be done for web spam detection. For example, spammers may use different tricks in different spam pages. Can we find topical spam patterns? For adversarial information retrieval, it is interesting and useful to construct a spam search engine which can rank spam pages related to a query like "what are the spam pages related to digital camera and ranked high by most search engines?"

References

[1] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval.* ACM Press/Addison-Wesley, 1999.

[2] A. A. Benczur *et al.*. Spamrank: Fully automatic link spam detection. In *AIRWeb'05*.

[3] M. Brinkmeier. Pagerank revisited. *ACM Transactions on Internet Technology (TOIT)*, 6(3):282–301, 2006.

[4] C. Castillo *et al.*. A reference collection for web spam. *SIGIR Forum*, 40(2):11–24, 2006.

[5] D. Fetterly *et al.*. Spam, damn spam, and statistics: Using statistical analysis to locate spam web pages. In *WebDB'04*.

[6] D. Fetterly *et al.*. Detecting phrase-level duplication on the world wide web. In *SIGIR'05*.

[7] Z. Gyöngyi *et al.*. Link spam detection based on mass estimation. In *VLDB'06*.

[8] Z. Gyöngyi and H. Garcia-Molina. Web spam taxonomy. In *AIRWeb'05*.

[9] Z. Gyöngyi *et al.*. Combating web spam with trustrank. In *VLDB'04*.

[10] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. In *SODA'98*.

[11] A. Ntoulas *et al.*. Detecting spam web pages through content analysis. In *WWW'06*.

[12] L. Page *et al.*. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.

[13] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[14] B. Wu and B. D. Davison. Identifying link farm spam pages. In *WWW'05*.

[15] B. Wu *et al.*. Topical trustrank: Using topicality to combat web spam. In *WWW'06*.

[16] B. Zhou. *Mining page farms and its application in link spam detection.* Master's thesis. School of Computing Science, Simon Fraser University, 2007.

[17] B. Zhou and J. Pei. Sketching landscapes of page farms. In *SDM'07*.