Robust PageRank and Locally Computable Spam Detection Features *

Reid Andersen
Microsoft Research, Redmond
reidan@microsoft.com

Christian Borgs
Microsoft Research, Redmond
borgs@microsoft.com

Jennifer Chayes
Microsoft Research, Redmond
jchayes@microsoft.com

John Hopcroft
Cornell University
jeh@cs.cornell.edu

Kamal Jain
Microsoft Research, Redmond
kamalj@microsoft.com

Vahab Mirrokni
Microsoft Research, Redmond
mirrokni@microsoft.com

Shanghua Teng
Boston University
steng@cs.bu.edu

ABSTRACT

Since the link structure of the web is an important element in ranking systems on search engines, web spammers widely use the link structure of the web to increase the rank of their pages. Various link-based features of web pages have been introduced and have proven effective at identifying link spam. One particularly successful family of features (as described in the SpamRank algorithm), is based on examining the sets of pages that contribute most to the PageRank of a given vertex, called *supporting sets*. In a recent paper, the current authors described an algorithm for efficiently computing, for a single specified vertex, an approximation of its supporting sets. In this paper, we describe several link-based spam-detection features, both supervised and unsupervised, that can be derived from these approximate supporting sets. In particular, we examine the size of a node's supporting sets and the approximate l_2 norm of the PageRank contributions from other nodes. As a supervised feature, we examine the composition of a node's supporting sets. We perform experiments on two labeled real data sets to demonstrate the effectiveness of these features for spam detection, and demonstrate that these features can be computed efficiently. Furthermore, we design a variation of PageRank (called Robust PageRank) that incorporates some of these features into its ranking, argue that this variation is more robust against link spam engineering, and give an algorithm for approximating Robust PageRank.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval models; I.2.6 [**Learning**]: Concept learning; G.2.2 [**Graph Theory**]: Graph labeling

General Terms

Theory, algorithms, performance

Keywords

Link spam, Local Algorithms, Graph Algorithms, Unsupervised learning, Directed Graphs, PageRank

1. INTRODUCTION

Web spam refers to attempts to increase the ranking of a web page by manipulating the content of a page and the link structure around a page. Web spam can decrease the quality of search results substantially, as well as increasing the cost of crawling, indexing, and storage in the search engine. As a result, identifying web spam is one of the top challenges in web search engines [11].

There are numerous approaches for detecting web spam, which may be based on web page content, link structure, or a combination of these. Successful techniques include the application of machine learning techniques to link-based features [2], the analysis of page content [14, 13], TrustRank [9] and Anti-TrustRank [15], statistical analysis of various page features [6], and transductive link spam detection [18]. One successful technique for spam detection has been to identify the sets of nodes that contribute the most to the PageRank of a node. Intuitively, this information is useful because a spam page may receive fairly large contributions from a relatively small number of pages in an engineered link farm, without receiving the same number of small contributions as a legitimate popular page. The analysis of supporting sets and PageRank contributions has been used as a tool for spam detection in the SpamRank algorithm of Benczur et al. [3], in the Spam Mass algorithm of Gyongyi et al. [8], and in recent work by Zhou [17].

The notion of supporting sets can be made precise by rigorously defining the contribution that a node u makes to the PageRank of a vertex v in terms of personalized PageRank. For a webgraph $G = (V, E)$ and a *teleportation constant* α (sometimes called the restart probability), let $\mathbf{PRM}_\alpha$ be

*This work was done while, John Hopcroft and Shang-Hua Teng were visiting Microsoft Research.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AIRWeb '08, April 22, 2008 Beijing, China.
Copyright 2008 ACM XXX-X-XXXXX-XXX-X ...$5.00.

the matrix whose u^{th} row is the personalized PageRank vector of u. The PageRank contribution of u to v, written $\mathrm{pr}_\alpha(u \to v)$, is defined to be the entry (u, v) of this matrix. The PageRank of a vertex v is the sum of the vth column of the matrix $\mathbf{PRM}_\alpha$, and thus the PageRank of a vertex can be viewed as the sum of the contributions from all other vertices. The *contribution vector* of v is defined to be the vth column of the matrix $\mathbf{PRM}_\alpha$, whose entries are the contributions of every vertex to the PageRank of v. The δ-supporting set of a vertex is defined to be the set of vertices that contribute to v at least a δ-fraction of the PageRank of the vertex v.

In a recent paper, the current authors introduced an algorithm for approximating the contribution vector and supporting sets for a single specified vertex [1]. This algorithm can produce a reasonable approximation of the contribution vector for a specified vertex, while adaptively examining only a small number of vertices near the vertex. This is analogous to the way that a single personalized PageRank vector can be approximated efficiently by examining only a small portion of the input graph (see for example [7, 12, 4, 16]). This is different from the methods used to compute contribution vectors in existing spam detection algorithms (for example [3]), where one computes an approximation of each personalized PageRank vector in the graph, makes these vectors the columns of an approximate PageRank matrix, and then takes the approximate contribution vectors to be the columns of the transposed matrix.

In this paper, we define several spam-detection features based on locally computed approximate supporting sets. We use these features for both supervised and unsupervised learning approaches for spam detection. For supervised learning, we use some prelabeled data to discover the labels of the unlabeled nodes. In particular, we examine the composition of labels of a node's supporting sets to predict its spam score. For unsupervised learning, we define several features based on characteristics of a node's supporting set. For example, we examine the size of a node's supporting sets, and the approximate l_2 norm of the PageRank contributions from other nodes. We perform experiments on a benchmark spam detection data set that demonstrate the effectiveness of these features for spam detection, and perform experiments on a larger domain graph dataset to study the computational cost of computing these features.

Finally, we define a variation of PageRank called *Robust PageRank*, which we argue is more robust against link spam engineering. In Robust PageRank, each vertex is allowed to contribute at most a certain amount to the PageRank of another node. We show that Robust PageRank can be approximated efficiently by combining PageRank with several of the features we consider. This ranking system is related to the Truncated PageRank algorithm studied by [2], in which the contribution from nodes of distance less than d is ignored. Since a node at a larger distance may contribute more to the PageRank of a given node than a nearby node, robust PageRank can be viewed as a refinement of Truncated PageRank.

Organization.

This paper will be organized as follows. In Section 2, we review necessary background, including PageRank, personalized PageRank, PageRank contribution vectors, and the algorithm for computing the supporting sets of a specified vertex that was given in [1]. In Section 4, we define several locally computable spam detection features. In Section 5, we present experimental results on two real data sets. In Section 6, we define a refined notion of PageRank that we call the *Robust PageRank*. We describe an algorithm for approximating Robust PageRank, and give some experimental results to justify the use of Robust PageRank.

2. PRELIMINARIES

In this section we explain the algorithmic tools used in our experiments. In particular, we describe as briefly as possible (without proofs or analysis) the algorithm for approximating PageRank contribution vectors that appeared in [1].

The web can be modeled by a directed graph $G = (V, E)$ where V are web pages and a directed edge $(u \to v) \in E$ represents a hyperlink in u that references v. To deal with the problem of dangling nodes with no out-edges, we assume an artificial node with a single self-loop has been added to the graph, and an edge has been added from each dangling node to this artificial node. Let A denote the adjacency matrix of G. For each $u \in V$, let $d_{out}(u)$ denote the out-degree of u and let $d_{in}(u)$ denote the in-degree of u. Let D_{out} be the diagonal matrix of out-degrees.

We will now define PageRank vectors and contribution vectors. For convenience, we will view all vectors as row vectors, unless explicitly stated otherwise.

For a teleportation constant α, the PageRank vector $\mathbf{pr}_\alpha$ defined by Brin and Page [5] satisfies the following equation:

$$\mathbf{pr}_\alpha = \alpha \cdot \mathbf{1} + (1 - \alpha) \cdot \mathbf{pr}_\alpha \cdot M, \tag{1}$$

where M is the random walk transition matrix given by $M = D_{out}^{-1} A$ and $\mathbf{1}$ is the row vector of all 1's (always of proper size). The PageRank of a page u is then $\mathrm{pr}_\alpha(u)$. When there is no danger of confusion, we may drop the subscript α. Note that the above definition corresponds to the normalization $\sum_u \mathrm{pr}_\alpha(u) = |V|$.

Similarly, the personalized PageRank vector $\mathbf{ppr}(\alpha, u)$ of a page $u \in V$, defined by Haveliwala [10], satisfies the following equation.

$$\mathbf{ppr}(\alpha, u) = \alpha \cdot \mathbf{e}_u + (1 - \alpha) \cdot \mathbf{ppr}(\alpha, u) \cdot M, \tag{2}$$

where $\mathbf{e}_u$ is the row vector whose u^{th} entry is equal to 1.

Let $\mathbf{PRM}_\alpha$ denote the matrix whose uth row is the personalized PageRank vector $\mathbf{ppr}(\alpha, u)$. The (global) PageRank vector $\mathbf{pr}_\alpha$ is then $\mathbf{1} \cdot \mathbf{PRM}_\alpha$, the sum of all the personalized PageRank vectors. The *PageRank contribution* of u to v is defined to be the (u, v)th entry of $\mathbf{PRM}_\alpha$, and will be written $\mathrm{ppr}_\alpha(u \to v)$. The contribution vector $\mathbf{cpr}(\alpha, v)$ for the vertex v is defined to be the row vector whose transpose is the vth column of $\mathbf{PRM}_\alpha$. If $\mathbf{c} = \mathbf{cpr}(\alpha, v)$ is the contribution vector for v, then we denote by $\mathbf{c}(S)$ the total contribution of the vertices in S to the PageRank of v. In particular, we have $\mathbf{c}(V) = \mathrm{pr}_\alpha(v)$ and $\mathbf{c}(u) = \mathrm{ppr}_\alpha(u \to v)$.

3. LOCAL APPROXIMATION OF CONTRIBUTION VECTORS

In this section, we present an algorithm for computing an approximation of the contribution vector $\mathbf{c} = \mathbf{cpr}(\alpha, v)$ of a vertex v. We will derive spam detection features from these approximate contribution vectors.

Definition 1. A vector $\tilde{\mathbf{c}}$ is an *ϵ-absolute-approximation* of the contribution vector $\mathbf{c} = \mathbf{cpr}(\alpha, v)$ if $\tilde{\mathbf{c}} \geq 0$ and, for all

vertices u,

$$\mathbf{c}(u) - \epsilon \leq \tilde{\mathbf{c}}(u) \leq \mathbf{c}(u).$$

The *support* of a non-negative vector $\tilde{\mathbf{c}}$, denoted $\mathrm{Supp}(\tilde{\mathbf{c}})$, is the set of all vertices whose entries in $\tilde{\mathbf{c}}$ are strictly positive. The vector $\mathbf{c}$ has a canonical ϵ-absolute-approximation. Let $\bar{\mathbf{c}}$ denote the vector

$$\bar{\mathbf{c}}(u) = \begin{cases} \mathbf{c}(u) & \text{if } \mathbf{c}(u) > \epsilon \\ 0 & \text{otherwise} . \end{cases}$$

Clearly, $\bar{\mathbf{c}}$ is the ϵ-absolute-approximation of $\mathbf{c}$ with the smallest support. Moreover, $\|\bar{\mathbf{c}}\|_1 \leq \|\mathbf{c}\|_1$ and therefore $|\mathrm{Supp}(\bar{\mathbf{c}})| \leq \|\mathbf{c}\|_1/\epsilon$. Our local algorithm attempts to find an approximation $\tilde{\mathbf{c}}$ of $\mathbf{c}$ which has a similar support structure to that of $\bar{\mathbf{c}}$.

The theorem below describes `ApproxContrib`, our algorithm for computing an ϵ-absolute-approximation of the contribution vector of a target vertex v.

We give an upper bound on the number of vertices examined by the algorithm that depends on $\mathrm{pr}_\alpha(v)$, ϵ, and α, but is otherwise independent of the number of vertices in the graph. The algorithm performs a sequence of operations, which we call pushback operations. Each pushback operation is performed on a single vertex of the graph, and requires time proportional to the in-degree of that vertex. We place an upper bound on the *number* of pushback operations performed by the algorithm, rather than the total running time of the algorithm. The total running time of the algorithm depends on the in-degrees of the sequence of vertices on which the pushback operations were performed. The number of pushback operations is an upper bound on the number of vertices in the support of the resulting approximate contribution vector.

THEOREM 1. *The algorithm* `ApproxContrib`(v, α, ϵ) *has the following properties. The input is a vertex v and two constants α and ϵ in the interval $(0, 1]$. The algorithm computes a vector $\tilde{\mathbf{c}}$ such that $0 \leq \tilde{\mathbf{c}} \leq \mathbf{c}$, and $\tilde{\mathbf{c}}$ is an ϵ-absolute approximation of $\mathbf{cpr}(\alpha, v)$. The number of pushback operations performed by the algorithm is at most $1 + \mathrm{pr}_\alpha(v)/\alpha\epsilon$.*

The algorithm and its analysis are described in [1]. In this paper, we will treat the algorithm as a black box for efficiently computing approximate contribution vectors for specified vertices. In the following section, we describe spam detection features that can be derived from these approximate contribution vectors.

4. LOCALLY COMPUTABLE FEATURES

In this section, we formally define our locally computable spam detection features. We mainly use the local algorithm for approximating contribution vectors to compute approximate supporting sets, sets of vertices that contribute significantly to the PageRank of a target vertex. First we define approximate contributing sets that are useful in the definition of most of our features.

4.1 Approximate contributing sets

We can use the algorithm for computing approximate contribution vectors to compute an approximation of the set of vertices that are significant contributors to the PageRank of v. Given a vertex v, we define the following contributing sets.

- **δ-significant contributing set**: The set

$$S_\delta(v) = \{u \mid \mathrm{ppr}_\alpha(u \to v) > \delta \cdot \mathrm{pr}_\alpha(v)\}.$$

- **ρ-supporting set**: any set S of pages such that

$$\mathrm{ppr}_\alpha(S \to v) \geq \rho \cdot \mathrm{pr}_\alpha(v).$$

4.2 Unsupervised learning features

We define several features that are only based on the link structure and are not based on any labeled data. We show that these features can be computed locally and show experimental results for their performance. Here is a list of our locally computable features:

1. **Size of δ-significant contributing set.** For a node v, this feature is simply $|S_\delta(v)| = |\{u \mid \mathrm{ppr}_\alpha(u \to v) > \delta \cdot \mathrm{pr}_\alpha(v)\}|$. We expect to observe that the contribution set of spam nodes is significantly smaller than the size of contribution set of non-spam nodes.

2. **Contribution from vertices in the δ-significant contributing set.** This feature for node v is equal to $\sum_{u \in S_\delta(v)} \frac{\mathrm{ppr}_\alpha(u \to v)}{\mathrm{pr}_\alpha(v)}$.

3. **l_2 norm of δ-significant contributing vector.** This feature for node v is equal to $\sum_{u \in S_\delta(v)} \left(\frac{\mathrm{ppr}_\alpha(u \to v)}{\mathrm{pr}_\alpha(v)}\right)^2$. The special case of this vector with $\delta = 0$ is denoted by L2Norm, and is equal to $\sum_{u \in V(G)} \left(\frac{\mathrm{ppr}_\alpha(u \to v)}{\mathrm{pr}_\alpha(v)}\right)^2$.

As a base for comparison, we also report the performance of some simple unsupervised features as follows:

1. The indegree of each node (Indegree).

2. The ratio of the PageRank of a node and its indegree (PRIndegree).

4.3 Approximate computation of features

We can use the algorithm for computing approximate contribution vectors to compute approximations of these spam detection features. To compute approximate versions of the features with parameter δ, we will compute an ϵ-approximate contribution vector $\tilde{\mathbf{c}}$ with $\epsilon = \delta \cdot \mathrm{pr}_v$. We let $\tilde{S}_\delta$ be the set of vertices with value at least $\delta \cdot \mathrm{pr}_v$ in the vector $\tilde{\mathbf{c}}$.

The set $\tilde{S}_\delta$ can be computed using $1/\alpha\delta + 1$ `pushback` operations, from Theorem 1. This set provides a good approximation to the set of vertices that are significant contributors to the PageRank of v, in the sense that $S_{2\delta} \subseteq \tilde{S}_\delta \subseteq S_\delta$. We define the following approximate features.

1. **(Approximate) size of the δ-significant contributing set (SuppSizeDelta).** To approximate this feature for a node v, we compute

$$|\tilde{S}_\delta| = |\{u \mid \tilde{\mathbf{c}}(u) > \delta \cdot \mathrm{pr}_\alpha(v)\}|.$$

2. **(Approximate) contribution from δ-significant contributing set (ContributePercent).** For a node v, we compute

$$\sum_{u \in \tilde{S}_\delta} \frac{\tilde{\mathbf{c}}(u)}{\mathrm{pr}_\alpha(v)}.$$

3. **(Approximate) l_2 norm of contribution vector (L2NormDelta).** For a node v, we compute

$$\sum_{u \in V(G)} \left(\frac{\tilde{\mathbf{c}}(u)}{\mathrm{pr}_\alpha(v)} \right)^2 .$$

4.4 Supervised learning features

In supervised learning, we use a set of already labeled nodes and predict the label of unlabeled nodes. As before, we compute an approximation $\tilde{\mathbf{c}}$, and let $\tilde{S}_\delta(v)$ be the approximate supporting set. We let $\tilde{T}_\delta(v)$ be the set of nodes in $\tilde{S}_\delta(v)$ that are labeled as spam. Here is a list of our locally computable supervised learning features:

1. **Fraction of nodes in the supporting set labeled as spam (SupervisedUnweighted).** For a node v, Then this feature is equal to $|\tilde{T}_\delta(v)|/|\tilde{S}_\delta(v)|$.

2. **Contribution from nodes in the supporting set labeled as spam (SupervisedWeighted).** For a node v, this feature is equal to $\sum_{u \in \tilde{T}_\delta(v)} \tilde{\mathbf{c}}(u) / \sum_{u \in \tilde{S}_\delta(v)} \tilde{\mathbf{c}}(u)$, the (approximate) contribution from nodes in the supporting set labeled as spam, divided by the (approximate) contribution from all nodes in the supporting set.

We remark that the feature (SupervisedWeighted) is essentially an approximation of the feature obtained by computing PageRank with the starting vector set to be 1 on the nodes labeled as spam. Ideas similar to this feature has been considered before as a way of propagating spam scores, for example in the Spam Mass algorithm [8]. Here, the novelty is that the value of the feature may be efficiently computed for a single specified vertex.

As a base for comparison, we also report the performance of the following simple supervised feature:

1. (SupervisedIndegree) The ratio between the number of spam nodes in the incoming neighbors of a node and the indegree.

5. EVALUATION OF FEATURES

In this section, we describe our experimental results along with some observations about the two real data sets that we used for spam detection.

5.1 Data Sets

The main data set on which we evaluate the performance of our features is a part of th UK host graph studied by Becchetti et al. [2] (`UKhost`). This graph has 11402 nodes and average degree of 65. Almost half of nodes of this graph is human-labeled for spam and non-spam nodes. As reported by Becchetti et al. [2], 5750 of the hosts have been labeled manually. For each host, they inspected a few pages manually and marked it as spam if they found a link farm inside the host. As a result, 674 hosts are marked as spam, and 4947 hosts are marked as normal hosts.

For the purpose of evaluating the running time of our algorithms, we also run our algorithm on a second larger data set that is part of the domain graph created by MSN search in May of 2006 (`MSNdom`). This domain graph has around 55 million nodes and around 600 million links. The largest strongly connected component has around 13 million and 400 thousand nodes, and the largest connected component of the underlying undirected graph has around 44 million nodes. It is interesting to observe that among nodes that are not in the strongly connected component 21 million nodes have a directed path from the large strongly connected component, and around 6 million and 500 thousand nodes have a directed path to the large strongly connected component. The large number of nodes that are not in the strongly connected component indicate that many domains not in the strongly connected component manage to get a link from a node in this component.

5.2 Running Time

We run the local algorithm for finding the δ-significant contributing set on a set of high PageRank nodes on both of the data sets. For the UK host graph, we run the algorithm for the 24% highest PageRank nodes in the graph (i.e., around 2800 nodes). For the MSN domain graph, we run the algorithm for 5000 nodes randomly chosen from the 10% high PageRank nodes. In Table 1, we report the average size of the δ-significant contributing sets, the average number of pushback operations, and the average number of total operations performed by the local algorithm for these data sets.

For a given δ, the theoretical bound for the number of pushback operations is $O(\frac{1}{\alpha\delta})$. The total running time of the algorithm is also related to the sum of the indegree of the nodes from which we perform a pushback operation. For both of our data sets, we observe that the number of pushback operations is in fact much smaller than the worst case theoretical bound. We also note that for the larger MSN domain graph, the local algorithm has a better running time.

5.3 Experimental Results

In this section, we report the performance of our features on the UK data set. Since the problem of spam detection is more critical and important for high PageRank nodes, we evaluate the features only on a set of nodes with high PageRank scores. In particular, we take the 24% of the nodes with the highest PageRank to call the "high PageRank nodes".

Let spam be the set of high PageRank labeled spam nodes, and nonspam be the set of high PageRank labeled nonspam nodes. Based on each feature f, we define a classifier that reports a subset of nodes as spam and the rest of nodes as nonspam. This classifier works based on a threshold $\mathsf{threshold}_f$ as follows: all nodes v for which $f(v) \leq \mathsf{threshold}_f$ are in one class and the rest of nodes are in the other class. For each feature f, let reportedspam be the set of high PageRank nodes reported as spam. For each feature f, we consider the following parameters:

1. **Average value ($\bar{f}$).** This parameter is equal to the average value this feature on a set of high PageRank nodes. For the UK host graph we take the average over the 24% high PageRank nodes.

2. **Average value on spam ($\bar{f}(\mathsf{spam})$).** This parameter is equal to the average value this feature on a set of high PageRank spam nodes.

3. **Average value on non-spam ($\bar{f}(\mathsf{nonspam})$).** This parameter is equal to the average value this feature on a set of high PageRank non-spam nodes.

4. **Percentage of false positives (FalsePos).** This metric is equal to the percentage of nonspam nodes that are reported as spam over the whole number of nonspam nodes, $\frac{\mathsf{nonspam} \cap \mathsf{reportedspam}}{\mathsf{nonspam}}$.

5. **Percentage of false negatives or Recall (Rec).** This metric is equal to the percentage of spam nodes that are reported as spam over the whole number of spam nodes, $\frac{\mathsf{spam} \cap \mathsf{reportedspam}}{\mathsf{spam}}$.

6. **Precision (Prec).** This metric is equal to the percentage of spam nodes among the total number of reported spam nodes over the total number of spam and nonspam nodes that are reported as spam nodes, $\frac{\mathsf{spam} \cap \mathsf{reportedspam}}{(\mathsf{nonspam} \cup \mathsf{spam}) \cap \mathsf{reportedspam}}$.

A better feature has a smaller FalsePos and larger Rec and Prec. In particular, as verifying if a node is spam or not is very costly, we do not want to report many nonspam nodes as spam. It is also clear that there is a tradeoff among these parameters. As a result, for each feature f, we set the threshold $\mathsf{threshold}_f$ for the classifier such that the parameter FalsePos is bounded above by 5% (or 2%). If smaller FalsePos is desirable, we also report a percentile chart for a subset of features by which one can figure out the threshold $\mathsf{threshold}_f$ that results to a better FalsePos. In each *percentile chart*, we sort the feature for nodes, and divide all the high PageRank nodes into 10 classes with the same number of nodes, and report the number of spam, nonspam, and unlabeled nodes in each class. The unsupervised learning experimental results on the UK graph are summarized in table 2. We also report the performance of the base features Indegree and PRIndegree. Percentile charts for features SuppSizeDelta and ContributePercent for $\delta = 10^{-4}$, and SuppSizeDelta and L2NormDelta for $\delta = 10^{-5}$ is depicted in Figures 1,2,4, and 3. As we expect, the supervised learning features perform slightly better than the unsupervised learning approaches. The results of all supervised learning features are summarized in Table 3.

5.4 Observations

In this section, we summarize some of our observations about the experimental results. For each locally computable feature with a parameter δ, the running time of the algorithm depends on the number of pushback operations or the size of the δ-significant contribution set. For the UK host graph, the size of δ-significant contribution set for $\delta = 10^{-2}, 10^{-3}, 10^{-4}$, and 10^{-5} are around 4, 25, 305, and 5700, respectively, for high PageRank nodes. Among the studied unsupervised learning features the size of supporting set SuppSizeDelta and ContributePercent perform the best. The performance of L2NormDelta feature is reasonable but significantly worse than the former two features. These features outperform the base unsupervised learning features Indegree, and PRIndegree significantly. For $\delta = 10^{-4}$ for which the size of the supporting set is very small (around 300), the best unsupervised learning feature is ContributePercent. Interestingly, the performance of these unsupervised learning features is slightly better than the base supervised learning feature SupervisedIndegree. As expected, the supervised learning feature SupervisedUnweighted perform slightly better than all of the unsupervised learning features. Finally, we note that for all the above features, the performance of the feature increase as the parameter δ decreases.

Finally, we observe that the unsupervised link spam detection methods developed here are only useful for spam detection over high PageRank nodes. In fact, we performed some experiments for all nodes of the graph, and the results are much worse than the results presented in this paper for high PageRank nodes. For example, by setting the parameter FalsePos to 0.02 and 0.05 respectively, the parameter Rec changes from 0.10 to 0.23, and from 0.05 to 0.15. As a result, we do not report these numbers, and emphasize that the application of these features are in detecting spamminess of high PageRank nodes. Note that the link spam detection problem is more crucial for high PageRank nodes as they may appear in the top positions in answering queries. The intuition behind the bad performance of these features for low PageRank nodes is that the link structure around many low PageRank nodes is not rich enough to imply any information about the spamminess of these nodes.

6. ROBUST PAGERANK

In this section, using the ideas of contribution sets, we design a ranking system, called the *Robust PageRank*, as a refinement to the PageRank algorithm. This system is more reliable against link spam in that the link spammers should invest more in buying new domains to increase the rank of a node in this system. We will support this claim by some experimental results.

The PageRank of a node v can be written as the sum of contribution of other nodes to v. Formally,

$$\mathrm{pr}_\alpha(v) = \sum_{u \in V(G)} \mathbf{ppr}(u, v).$$

The idea behind robust PageRank is to decrease the effect of the most influential nodes on the PageRank of a node to make the ranking system more robust against link spam engineering. By decreasing the effect of a link spammer who gets the main portion of his PageRank from a link farm around him (and not from a variety of many nodes), we expect that the link spammer lose a large portion of his PageRank. This idea leads to the following definition of Robust PageRank. This can be done by decreasing the contribution of nodes with a contribution more than a threshold δ to δ. Formally, for a threshold δ, we define the robust PageRank of a vertex v, $\mathrm{Robustpr}^\delta_\alpha(v)$, as follows:

$$\mathrm{Robustpr}^\delta_\alpha(v) = \sum_{u \in V(G)} \min(\mathbf{ppr}(u, v), \delta).$$

Since we decrease the contribution of a few most influential nodes to the PageRank of a node, we decrease the PageRank of spam nodes more than the nonspam nodes. This will make the new ranking system more robust. We next describe how to approximate the robust PageRank of nodes by computing approximate contribution vectors, and then give some experimental evidence on a data set to support our claim.

We compute an approximation of the Robust PageRank in the following way. First, we can rewrite the Robust PageR-

ank as follows.

$$\begin{aligned}\mathrm{Robustpr}_\alpha^\delta(v) &= \sum_{u\in V(G)} \min(\mathbf{ppr}(u,v),\delta)\\ &= \sum_{u\in V(G)} \mathbf{ppr}(u,v) - \sum_{u\in S_\delta(v)} (\mathbf{ppr}(u,v)-\delta)\\ &= \mathrm{pr}_\alpha(v) - \sum_{u\in S_\delta(v)} \mathbf{ppr}(u,v) - \delta|S_\delta(v)|.\end{aligned}$$

To compute an approximation of the Robust PageRank for a given vertex v, we compute an approximate contribution vector $\tilde{\mathbf{c}}$ for v with error $\epsilon = \delta \mathrm{pr}_\alpha(v)$, and replace the PageRank contributions $\mathbf{ppr}(u,v)$ by the approximate contributions $\tilde{\mathbf{c}}(u)$. We then define the following approximate notion of Robust PageRank, $\mathrm{ApproxRobustpr}_\alpha^\delta(v) = \mathrm{pr}_\alpha(v) - \sum_{u\in\tilde{S}_\delta(v)} \tilde{\mathbf{c}}(u) - \delta|\tilde{S}_\delta(v)|$.

6.1 Experiments with Robust PageRank

Figures 7 and 8 depict the percentile chart of PageRank and robust PageRank of the 24% highest PageRank nodes of the UK host graph ordered in the decreasing order of their PageRank. By comparing these two percentile charts, it is clear that a big portion of spam nodes are pushed to lower ranks in the robust PageRank system.

In order to complement the above charts and give more quantitative evidence to support our claim about the effectiveness of robust PageRank compared to the PageRank vector, we also evaluate the ratio between the robust PageRank and PageRank of nodes and use it as a feature, denoted by NormalizedRobustPR. For a node v this feature is formally

$$\mathsf{NormalizedRobustPR} = \frac{\mathrm{ApproxRobustpr}_\alpha^\delta(v)}{\mathrm{pr}_\alpha(v)}.$$

As an evidence for the performance of robust PageRank, we show that this feature results in a good classifier for spam detection. The metrics for this classifier is reported in Table 4. The results show that that the decrease in robust PageRank compared to PageRank is significantly more for spam nodes. Despite the other studied features, the quality of this feature does not increase as we decrease δ. In particular, the quality of this feature is the best for $\delta = 10^{-3}$. For this $\delta = 10^{-3}$, this feature outperforms all the previous unsupervised and supervised learning approaches.

7. CONCLUSIONS

In this paper, we explore a set of features for link spam detection that can be computed locally by approximating PageRank contribution vectors. Most of the features have a parameter δ for the desired approximation factor of the contribution set. In order to find an appropriate tradeoff between the running time of the algorithm and the performance of the spam detection features, a suitable parameter δ should be computed based on the data set. Among the unsupervised learning features, the features ContributePercent, NormalizedRobustPR, and SuppSizeDelta perform quite well for spam detection. For the UK host graph, among unsupervised features, the feature NormalizedRobustPR performs the best for $\delta = 10^{-3}$, and the feature ContributePercent performs the best for $\delta = 10^{-4}$. In particular, these features perform even better than the supervised learning feature SupervisedIndegree. The supervised learning feature SupervisedUnweighted has the best performance among all features.

8. REFERENCES

[1] R. Andersen, C. Borgs, J. Chayes, J. Hopcroft, V. Mirrokni, and S. Teng. Local computation of PageRank contributions. In *5th International Workshop of Algorithms and Models for the Web-Graph*, 2007.

[2] L. Becchetti, C. Castillo, D. Donato, S. Leonardi, and R. Baeza-Yates. Link-based characterization and detection of web spam. In *In Proceedings of the 2nd International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2006.

[3] A. Benczur, K. Csalogany, T. Sarlos, and M. Uher. Spamrank - fully automatic link spam detection. In *First International Workshop on Adversarial Information Retrieval on the Web*, 2005.

[4] P. Berkhin. Bookmark-coloring algorithm for personalized PageRank computing. *Internet Math.*, 3(1):41–62, 2006.

[5] 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.

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

[7] D. Fogaras and B. Racz. Towards scaling fully personalized PageRank. In *Proceedings of the 3rd Workshop on Algorithms and Models for the Web-Graph (WAW)*, pages 105–117, October 2004.

[8] Z. Gyöngyi, P. Berkhin, H. Garcia-Molina, and J. Pedersen. Link spam detection based on mass estimation. In *Proceedings of the 32nd International Conference on Very Large Databases*. ACM, 2006.

[9] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with trustrank. In *VLDB*, pages 576–587, 2004.

[10] T. H. Haveliwala. Topic-sensitive PageRank: A context-sensitive ranking algorithm for web search. *IEEE Trans. Knowl. Data Eng.*, 15(4):784–796, 2003.

[11] M. Henzinger, R. Motwani, and C. Silverstein. Challenges in web search engines. In *In: Proc. of the 18th International Joint Conference on Artificial Intelligence*, pages 1573–1579, 2003.

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

[13] G. Mishne and D. Carmel. Blocking blog spam with language model disagreement, 2005.

[14] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly. Detecting spam web pages through content analysis. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 83–92, New York, NY, USA, 2006. ACM Press.

[15] R. Raj and V. Krishnan. Web spam detection with anti-trust rank. In *Proc. of the 2nd International Worshop on Adversarial Information Retreival on the Web*, pages 381–389, 2006.

[16] T. Sarlós, A. A. Benczúr, K. Csalogány, D. Fogaras, and B. Rácz. To randomize or not to randomize: space optimal summaries for hyperlink analysis. In *WWW*, pages 297–306, 2006.

[17] B. Zhou. Mining page farms and its application in link spam detection. *Master's Thesis, Simon Fraser University, 2007.*

[18] D. Zhou, C. Burges, and T. Tao. Transductive link spam detection. In *AIRWeb*, 2007.

Data Set	δ	Size of δ-Supp Set	#Operations	#Pushbacks	Theory Bound
UKhost	10^{-2}	4.08	2589	61	1.3×10^4
UKhost	10^{-3}	25.89	32665	576	1.3×10^5
UKhost	10^{-4}	305.87	1071449	11115	1.3×10^6
UKhost	10^{-5}	5737.99	29800925	313649	1.3×10^7
MSNdom	10^{-2}	5.15	618	25	1.3×10^4
MSNdom	10^{-3}	28.54	6714	190	1.3×10^5
MSNdom	10^{-4}	197.75	67715	1682	1.3×10^6
MSNdom	10^{-5}	1747.38	692614	17410	1.3×10^7
MSNdom	10^{-6}	18672.24	9141900	241879	1.3×10^8

Table 1: Running Time for two data sets.

Feature	Rec_1	Prec_1	Rec_2	Prec_2	$\overline{f}$	$\bar{f}$(spam)	$\bar{f}$(nonspam)
Indegree	0.23	0.51	0.56	0.50	197.98	34.07	251.86
PRIndegree	0.19	0.48	0.50	0.47	0.03	0.08	0.02
SuppSizeDelta,$\delta = 10^{-2}$	0.61	0.72	0.70	0.55	4.08	11.13	2.18
SuppSizeDelta,$\delta = 10^{-3}$	0.20	0.46	0.21	0.27	25.89	29.55	21.98
SuppSizeDelta,$\delta = 10^{-4}$	0.30	0.56	0.92	0.62	305.87	82.86	364.29
SuppSizeDelta,$\delta = 10^{-5}$	0.22	0.49	0.97	0.63	5737.99	281.27	7321.71
ContributePercent,$\delta = 10^{-2}$	0.20	0.47	0.7	0.58	0.20	0.50	0.11
ContributePercent,$\delta = 10^{-3}$	0.32	0.58	0.95	0.63	0.29	0.69	0.17
ContributePercent,$\delta = 10^{-4}$	0.27	0.54	0.94	0.62	0.37	0.72	0.26
ContributePercent,$\delta = 10^{-5}$	0.32	0.58	0.97	0.63	0.70	0.73	0.69
L2NormDelta,$\delta = 10^{-2}$	0.18	0.44	0.64	0.05	0.02	0.05	0.01
L2NormDelta,$\delta = 10^{-3}$	0.17	0.43	0.72	0.56	0.02	0.06	0.01
L2NormDelta,$\delta = 10^{-4}$	0.17	0.43	0.72	0.56	0.02	0.06	0.01
L2NormDelta,$\delta = 10^{-5}$	0.17	0.43	0.72	0.564	0.02	0.06	0.01
L2Norm	0.17	0.43	0.72	0.56	0.11	0.22	0.08

Table 2: Performance of unsupervised learning features. Rec_1 **and** Prec_1 **correspond to** $\mathsf{FalsePos} = 0.02$**, and** Rec_2 **and** Prec_2 **correspond to** $\mathsf{FalsePos} = 0.05$**.**

Feature	Rec_1	Prec_1	Rec_2	Prec_2	$\overline{f}$	$\bar{f}$(spam)	$\bar{f}$(nonspam)
SupervisedIndegree	0.670	0.745	0.927	0.623	0.097	0.280	0.051
SupervisedUnweighted,$\delta = 10^{-2}$	0.090	0.275	0.110	0.164	0.051	0.318	0.009
SupervisedUnweighted,$\delta = 10^{-3}$	0.802	0.792	0.899	0.616	0.067	0.305	0.019
SupervisedUnweighted,$\delta = 10^{-4}$	0.841	0.811	0.963	0.633	0.069	0.233	0.025
SupervisedUnweighted,$\delta = 10^{-5}$	0.852	0.826	0.991	0.639	0.078	0.211	0.044
SupervisedWeighted,$\delta = 10^{-2}$	0.083	0.265	0.110	0.164	0.049	0.417	0.005
SupervisedWeighted,$\delta = 10^{-3}$	0.771	0.771	0.826	0.596	0.054	0.409	0.010
SupervisedWeighted,$\delta = 10^{-4}$	0.807	0.779	0.862	0.606	0.050	0.243	0.015
SupervisedWeighted,$\delta = 10^{-5}$	0.853	0.788	0.881	0.611	0.057	0.243	0.024

Table 3: Performance of supervised learning features. Rec_1 **and** Prec_1 **correspond to** $\mathsf{FalsePos} = 0.02$**, and** Rec_2 **and** Prec_2 **correspond to** $\mathsf{FalsePos} = 0.05$**.**

Feature	Rec_1	Prec_1	Rec_2	Prec_2	$\overline{f}$	$\bar{f}$(spam)	$\bar{f}$(nonspam)
NormalizedRobustPR, $\delta = 10^{-2}$	0.438	0.375	0.743	0.570	0.839	0.605	0.908
NormalizedRobustPR, $\delta = 10^{-3}$	0.853	0.695	0.963	0.633	0.730	0.333	0.851
NormalizedRobustPR, $\delta = 10^{-4}$	0.819	0.510	0.954	0.630	0.655	0.280	0.774
NormalizedRobustPR, $\delta = 10^{-5}$	0.789	0.510	0.945	0.628	0.352	0.265	0.376

Table 4: Performance of the feature NormalizedRobustPR**.** Rec_1 **and** Prec_1 **correspond to** $\mathsf{FalsePos} = 0.02$**, and** Rec_2 **and** Prec_2 **correspond to** $\mathsf{FalsePos} = 0.05$**.**

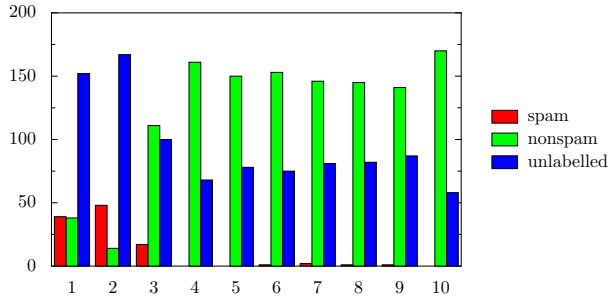

Figure 1: The percentile chart of feature SuppSizeDelta with $\delta = 10^{-4}$ ordered in the increasing order.

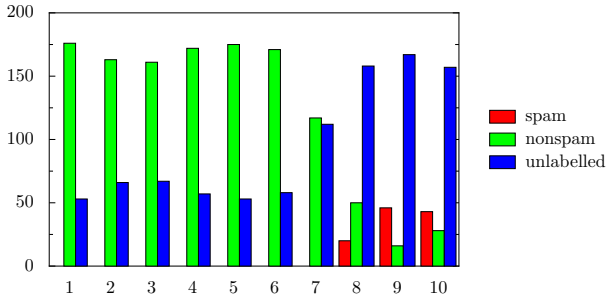

Figure 2: The percentile chart of feature ContributePercent with $\delta = 10^{-4}$ ordered in the increasing order.

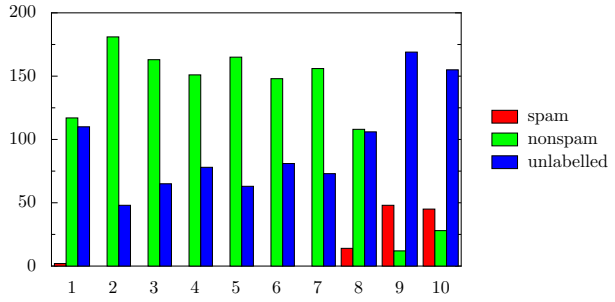

Figure 3: The percentile chart of the base feature ContributePercent with $\delta = 10^{-5}$ ordered in the increasing order.

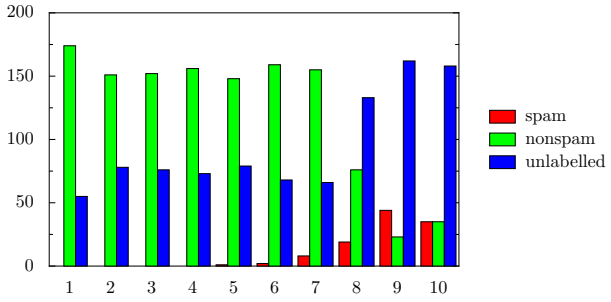

Figure 4: The percentile chart of feature L2NormDelta with $\delta = 10^{-4}$ ordered in the increasing order.

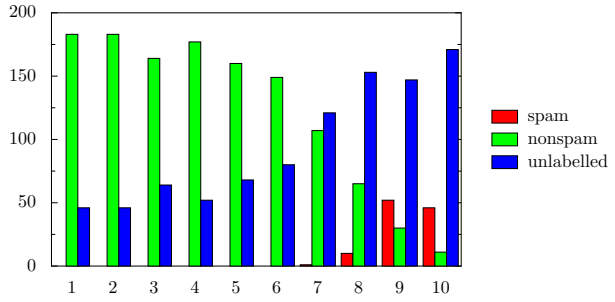

Figure 5: The percentile chart of SupervisedUnweighted with $\delta = 10^{-4}$.

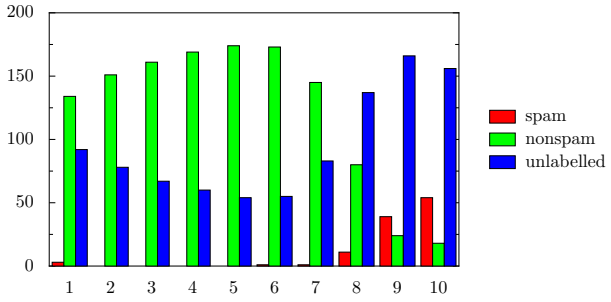

Figure 6: The percentile chart of SupervisedIndegree with $\delta = 10^{-4}$.

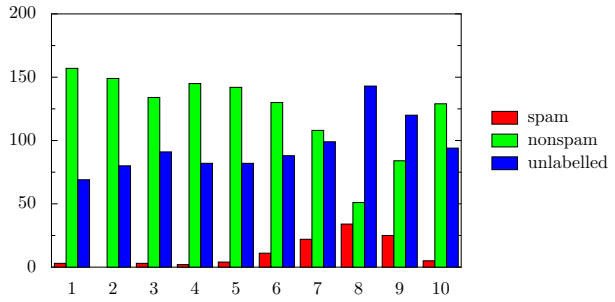

Figure 7: The percentile chart of PageRank vector ordered in the decreasing order.

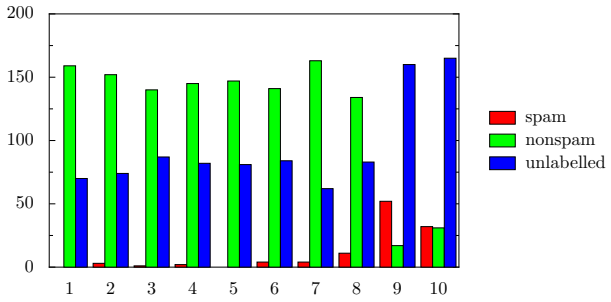

Figure 8: The percentile chart of robust PageRank vector with $\delta = 10^{-5}$ ordered in the decreasing order.