PerturbationRank: A Non-monotone Ranking Algorithm

Ye Du
University of Michigan
2260 Hayward Street
Ann Arbor, Michigan 48109
duye@umich.edu

James Leung
University of Michigan
2260 Hayward Street
Ann Arbor, Michigan 48109
jamleung@umich.edu

Yaoyun Shi
University of Michigan
2260 Hayward Street
Ann Arbor, Michigan 48109
shiyy@umich.edu

ABSTRACT

We introduce a new approach for ranking Web pages to capture the extent to which the whole Web depends on an individual Web page. The importance of a Web page is measured by how much the Web changes when the page is disconnected from the Web. While there are potentially many useful ways to quantify the change, in this work we focus on the following: represent the state of the Web by the output of a known ranking algorithm, and measure the change by an appropriate metric on the state space.

More specifically, we present PerturbationRank, as we have termed our class of ranking algorithms, in combination with PageRank and HITS. While both base algorithms, together with several others, view an incoming link as a positive endorsement and are in essence methods to account for the total endorsement, PerturbationRank represents a fundamental departure from this paradigm. As a consequence, it is able to capture usefulness signals missed previously.

We evaluate the performance of the new algorithms in comparison with the base algorithms, following the methodology of Borodin et al. (*ACM Transactions on Internet Technology*, **5**(1), 231–297, 2005). Our experiments demonstrate that the new and the base rankings are considerably correlated, have comparable effectiveness, and are at the same time substantially different. We conclude that PerturbationRank algorithms give alternative and useful rankings of the Web. Finally, we discuss how to improve the computation efficiency of PerturbationRank, as well as its other potential applications such as in network security and in combating spamdexing.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrival

General Terms

Algorithms, Evaluations

Keywords

PerturbationRank, Monotonicity

1. INTRODUCTION

The problem of ranking is ubiquitous in many fields of study, and forms the foundation of many large decision support systems. Web pages in particular present an interesting case, as the rapid growth of the Internet and the huge volume of search queries combine to make ranking Web pages both an important and difficult problem. Although there are several known rankings used with high effectiveness, it is impossible for a single ranking technique to survive scrutiny when tested against all possible measures of importance.

The Web has the advantage of allowing many orthogonal approaches for ranking, including the standard content-based relevance matching and document similarity methods. Additionally, because of the unique hypertext nature of the Web, link-based ranking algorithms are also an important component, in which the importance of a page is not decided by content of the page, but is a function of the linkage among pages. In the latter domain, HITS [15] and PageRank [19] are perhaps the two most well-known methods of ranking Web pages. As a result, they are also the most often studied link-based ranking techniques.

Most ranking algorithms including PageRank, HITS, the naive ranking by in-degrees, as well as a variant of HITS called SALSA [22], share the following property: if the set of pages pointing to a page p is a subset of those pointing to another page q, the ranking score of q is no less than that of p. This property, called *monotonicity* by Borodin et al. [22], reflects a fundamental aspect of those algorithms: each incoming link is considered a positive endorsement and the ranking is a way of accounting the total amount of endorsement. We argue, however, that such type of ranking is incomplete, and indeed, may fail to capture the intuitive notion of importance in some instances. For example, Page p may be the only page connecting to an otherwise disjoint portions of the Web, while Page q does not have any outgoing links. Monotone rankings will miss this aspect of the importance. An explicit construction of this example is illustrated in Figure 1. An additional drawback of a monotone ranking algorithm is that it allows a link spammer to easily boost the ranking of a target page by adding links to it. Therefore, it is of great interest to look for alternative ranking algorithms that are *not* monotone.

To this end, we present a new link-based approach, which we call PerturbationRank, to rank Web pages. PerturbationRank measures the importance of a Web page by the extent to which the Web as a whole depends on it. This dependence is further measured by the effect of disconnecting the page from the Web. The intuition is that, a page which, when disconnected, would cause a drastic change to the Web's state is most likely important, while if change is little, it is probably unimportant. Although there are potentially many ways to quantify the state of the Web and its changes, we take the following natural approach — use a known link-based ranking algorithm to represent the Web's state before and after the disconnection, and define the difference between states using an appropriate metric on the corresponding state space. In particular, we use both HITS and PageRank as the base Web ranking algorithms.

We will construct an example to show that unlike PageRank and HITS, the new ranking algorithm is not monotone. We also follow the methodology of Borodin et al. to evaluate the performances of the new algorithms and compare them to those of PageRank and HITS. More specifically, we run the ranking algorithms with a subset of queries used in [22] on the same data set, and evaluate the relevance of the top 10 return results using human-ranked results as the benchmark. Our experiments demonstrate that the new and the base rankings are considerably correlated, have comparable performances, and at the same time substantially different. We conclude that PerturbationRank gives an alternative and useful ranking of Web pages.

Since the publications of PageRank and HITS, there have been many subsequent works that improve, analyze, and extend both algorithms. Borodin et al. [22] and Langville et al. [16] provide excellent overviews of existing ranking algorithms. We will focus on PageRank and HITS in the next section. After defining PerturbationRank and its combinations with PageRank and HITS in Section 3, we evaluate its performance in Section 4. We then discuss the computational efficiency of PerturbationRank and its potential applications in other areas in Section 5. Finally we conclude in Section 6.

2. PAGERANK AND HITS

Since we will use PageRank and HITS to construct examples of PerturbationRank, we now describe in more details those two ranking algorithms. Denote the set of real numbers by $\mathbb{R}$. Recall that given a n-dimensional real vector $v = [v_i]_{i=1\ldots n} \in \mathbb{R}^n$, its ℓ_1 norm is $\|v\|_1 = \sum_{i=1}^n |v_i|$ and its ℓ_2 norm is $\|v\|_2 = \sqrt{\sum_{i=1}^n |v_i|^2}$.

The general idea behind the PageRank algorithm can be described as follows: if each page has an importance score, then it contributes to the importance score of other pages to which it links an amount based on its own importance. In turn, a page's importance is defined in terms of the importance of the pages which point to it. This encapsulates several scenarios which may be observed on the Web: first, a page is important if many thousands of other pages link to it, however small the importance of those linking pages may be individually: this is true, for example, of large gateways such as the Yahoo home page. Second, a page is important if it is linked to from a page which is itself important, even if the total number of in-links to the page is relatively low; this is true for those pages to which the Yahoo home page points.

Mathematically, PageRank may be realized thus: if $G = (V, E)$ is a graph, then let B_u for $u \in V$ be the set of nodes in V which point to u, and F_u be the set of nodes in V which are pointed to by u. Define the *PageRank* of the graph, R, to be [19]

$$R(u) = \sum_{v \in B_u} \frac{R(v)}{|F_v|} \tag{1}$$

Though this equation defines PageRank recursively, one can compute a steady-state ranking by fixing any initial total ranking, and then iterate this equation to update ranks for subsequent generations, until the computation converges; this is done by realizing the links between pages in a column-stochastic adjacency matrix, where the columns correspond to a given node's out-links; the final ranking is simply the principal eigenvector of the matrix, which may be computed by applying the matrix to any (non-degenerate) initial vector.[1]

Similar to PageRank, HITS is a link-based ranking algorithm which computes orthogonal, mutually defined scores, *hub* and *authority*, for pages in a given graph [15]. The graph, however, is not that of the entire Internet; rather, a 'root set' of up to a few hundred nodes is first chosen by taking the top results from a text-based search. This root set is then expanded by following its out-links and in-links, and bringing in those nodes to the subgraph. From this set of nodes, one represents the edges in an adjacency matrix A, where an entry $a_{i,j} = 1$ if the j^{th} node links to the i^{th} node. Authority and hub vectors, call them x and y respectively, are initialized to be 1 in every entry. Authority and hub values are updated in an iterative process by applying the matrix repeatedly, computing $x_{i+1} = Ay_i$ and $y_{i+1} = A^T x_i$ and re-normalizing x_{i+1} and y_{i+1} to have ℓ_2 norms equaling 1. This process is applied for a fixed number of iterations, or until the hub and authority score vectors converge to their steady-state values.

3. PERTURBATIONRANK

Intuitively, removing all the incoming and outgoing edges of an "important" node in a graph should disrupt the flow of the graph significantly; travellers following those edges along the graph can no longer take them. Similarly, if we remove all the incoming and outgoing edges of a node from a graph, the amount of disruption correlates to the node's importance in the graph. In the Web graph, removing all the incoming and outgoing links of more important nodes changes browsing traffic more than removing the links of less important nodes: for example, removal of all the incoming and outgoing links of an important gateway page should affect the reachability of a subgraph.

Consider a real world example like a group of traffic intersections, where each edge is a road and a vertex is an intersection of roads. Removal of all the incoming and outgoing

[1]For further details, see [5].

links of a Web page logically correlates to closing down an intersection. Very important intersections such as one in a dense city center that normally observe a high flow of cars will create a large disruption in the traffic graph because cars will now be forced to take alternative routes. Contrarily, low importance intersections such in rural areas where very few people live will alter traffic flow very little for the entire graph. We measure how much traffic changes on the other roads and use this value as a metric for determining the importance of an intersection.

We use a link-based ranking algorithm to give a sense of the graph's state before and after removal of links incident on a node. We begin by computing the initial ranking and then compute the ranking after removing all the incoming and outgoing edges of a node from the graph. We now measure the difference between the two rankings and make that the amount of disruption generated for the removed node. There are multiple methods of measuring this difference; they are discussed in section 3.3.

We formally define our ranking algorithm as a follows. Let $G = (V, E)$ be a directed multi-graph on n vertices. A *ranking* of (the vertices of) G is a function $\pi : V \to \mathbb{R}$. When there is no confusion, we may identify π with the vector $[\pi(v)]_{v \in V}$. Typically, π is normalized by some norm — e.g. π in PageRank is a probability distribution thus having a unit ℓ_1 norm while in HITS it is a nonnegative vector with unit ℓ_2 norm.

Let $v \in V$ and G_v be the graph obtained from G by removing all the incoming and outgoing edges of v. A *measurement of disruption* δ is a function that maps a ranking π of G and a ranking π_v of G_v to a non-negative real number $\delta(\pi, \pi_v)$. For example, we will use the following two measurements of disruption δ_{ℓ_1} and δ_{ℓ_2} that are based on the ℓ_1 norm and the ℓ_2 norm, respectively.

$$\delta_{\ell_1}(\pi, \pi_v) = \|\pi - \pi_v\|_1 . \qquad (2)$$

In plain English, δ_{ℓ_1} is designed for rankings that are probability distributions. It is precisely the ℓ_1 distance of the distribution π_v and the distribution π. Similarly, we define

$$\delta_{\ell_2}(\pi, \pi_v) = \|\pi - \pi_v\|_2 . \qquad (3)$$

Let r be a ranking algorithm and δ be a measurement of disruption. A *PerturbationRank* with the base ranking algorithm r and the measurement of disruption δ, denoted by $P_{r,\delta}$, is a ranking algorithm such that on $G = (V, E)$, and $v \in V$,

$$P_{r,\delta}(G)(v) = \delta(r(G), r(G_v)). \qquad (4)$$

A significant difference between PerturbationRank and base ranking algorithms, such as PageRank and the authority score of HITS, is that: in the base ranking algorithms, the ranking score of a node only depends on the set of nodes pointing to it. In PerturbationRank, the ranking score of a node depends not only on which set of nodes point to it, but also on which set of nodes it points to. As defined in before, the hub score of a node depends on which set of nodes it points to. However, HITS gives two separate ranking vectors. PerturbationRank provides one way to compose the authority and hubs scores given by HITS into one ranking vectors. In particular, PerturbationRank does not preserve certain basic property that the base ranking algorithms have. We will discuss this in details in subsection 3.4.

In the next two sub-sections, we describe PerturbationRank using PageRank and HITS as the base ranking algorithms. For each case, we will also describe the changes induced by the removal of a Web page and the intuition on which pages will gain or lose its ranking score.

3.1 PerturbationRank-PageRank

In this variant of PerturbationRank, we remove all the incoming and outgoing links of page p from the Web graph and compute PageRank on the perturbed graph. We then compute the difference between these rankings and assign it to be the perturbation value for page p. In particular, after removing all its incident links, page p would have the least PageRank score among all the pages in the graph. Furthermore, according to [23], the following upper bound is established for the perturbation value of page p,

$$\|\pi - \pi_p\|_1 \leq \frac{2(1-\epsilon)}{\epsilon} \sum_{i \in K} \pi_i \qquad (5)$$

where ϵ is the random jump factor and K is the set of pages perturbed by the removing of all the incoming and outgoing links of page p. Roughly speaking, the larger the size of K is, the greater the perturbation value of page p may have. Therefore, if page p has great indegree and outdegree, it tends to have large perturbation value.

The dynamics induced by the removal of all the incoming and outgoing links of p are rather complex. In the iterative application of the new transition matrix to the original stationary distribution, those pages that previously linked to p now distribute their PageRank values across one less outlink, increasing the PageRank apportioned to remaining outlinks. Similarly, those pages that were previously pointed to by p now receive less PageRank. Therefore, if p has a large PageRank then a distribution very close to p cannot be stable in this first iteration, and intuitively, p should have relatively large PerturbationRank as well. It is clear that if p's PageRank is tiny then the original PageRank is a good approximate stationary distribution of the new Markov chain, thus p is likely to have a small PerturbationRank as well. Therefore, there must be a certain correlation between PageRank and the PerturbationRank.

On the other hand, the changes in the first iteration will ripple through the whole graph and may be reversed in later iterations. Thus it does not appear a trivial task to predict the PageRank increase or decrease of a page, especially those forming a cycle with p, or being many steps away from p. This complex behavior is evidence that PerturbationRank-PageRank reflects a nontrivial and global property of a page.

Despite the complexity, there is one useful way to reason how the PageRank may change. A classical fact about Markov chain says (e.g. [13]) that the PageRank of a page q is precisely $1/t_q$, where t_q is the expected length of all cycles that start with and return to q. Under this formulation, we can look at the number of paths that pass through the page p in G: those paths no longer exist in G_p, which will affect q's

PageRank score. If the expected path length of the paths which pass through p is much higher than the expected path length of paths that do not pass through p, then t_q will tend to decrease and the PageRank score of q will tend to increase upon p's removal. Likewise, if the expected path length of the paths through p is much lower than the expected length of paths that do not pass through p, then t_q will increase and the PageRank score of q will decrease upon the removal of p's incoming and outgoing links.

3.2 PerturbationRank-HITS

In this variant of PerturbationRank, we first construct the HITS query-specific subgraph of the Web graph as per usual. We then remove all the incoming and outgoing links of page p from the HITS subgraph and examine the change in hub and authority values across the subgraph. We associate the change in hub and authority values with page p, and call them the authority and hub values of the PerturbationRank, respectively. In particular, page p should have both authority and hub score 0 after removing its incident links.

Let B be the set of pages that point to page p and F be the set of pages to which page p points. We explain why the PerturbationRank authority and hub values intuitively correlate with the original hub and authority values, respectively. Based on the "mutual reinforcement" formulation behind HITS, if the links of a page p with a high hub value is removed, the pages that p points to receive less endorsement from good hubs, thus tend to substantially lower their authority scores. If p has a very small hub value, it does not contribute much to the authority scores of pages it points to, thus removing it may not cause much change in the authority values. One also expects the correlation between PerturbationRank hub values and the original authority values to behave similarly.

We also expect that there should be correlations between the authority and the hub values of PerturbationRank. This is because those two rankings are related through a linear transformation. Thus a large change on the authority vector should cause a large change on the hub vector, and vice versa. This intuition may fail if the vector change happens to mostly lie in the null space of the linear transformation, but this appears unlikely for real data.

Just like in the PageRank case, the effect of the change also ripples through the whole graph. Thus we expect that our method captures more than the hub and authority scores.

3.3 Measurement of Disruption

There are many different methods available for measuring the difference of two rankings. We use two different types of measurements to evaluate PerturbationRank. In one method, we operate on the raw score vectors and compute the difference between the score vectors before and after the removal of all the incoming and outgoing links of a Web page. In the other, we simply examine the change in the ranking positions after ordering.

The first method applies to rankings that are normalized by certain norms. In particular, we may use δ_{ℓ_1} (defined in Eqn. 2) when the base ranking algorithm is PageRank, and use δ_{ℓ_2} (defined in Eqn. 3) when the base ranking algorithm is HITS.

The second method called *Kendall's* τ [14] measures the pairwise disagreement of the ranking vectors. Informally, if two vertices do not have the same relative ordering in both ranking vectors, then we increase Kendall's tau by 1. Formally it can be expressed as follows: Let τ_1 and τ_2 be two rankings, $K(\tau_1, \tau_2) = \sum_{i<j} \bar{K}_{i,j}(\tau_1, \tau_2)$ where $\bar{K}_{i,j}(\tau_1, \tau_2) = 0$ if element i and element j are ordered identically in τ_1 and τ_2, and $\bar{K}_{i,j}(\tau_1, \tau_2) = 1$ if element i and element j are different order in τ_1 and τ_2.

Kendall's tau is an egalitarian measurement in the sense that all (v_i, v_j) pairs contribute equally to the final score regardless of whether they appear in the top of the ranking vectors or the bottom. We instead weight the contributions to Kendall's tau based on the specific locations that the inversion occurs. Intuitively, a change higher up in the ranking should contribute more to the disruption. This adjustment makes sense in Web page ranking because a user pays more attention to the first page of results returned from a given search engine, so any change to the first page results should be weighted higher than a change in the second page of results, and so on.

We formally express this by treating ranking as a function $\tau : V \to \mathbb{Z}$, it takes a vertex and yields an integer representing the order of that vertex in the ranking. Let $\ell = min(\tau_1(i), \tau_1(j), \tau_2(i), \tau_2(j))$ and $0 < d \leq 1$. We call d the *damping factor*: it dampens the contribution of individual inversions to $K(\tau_1, \tau_2)$. We modify Kendall's tau to be

$$K^*(\tau_1, \tau_2) = \sum_{i,j} \bar{K}^*_{i,j}(\tau_1, \tau_2)$$

where

$$\bar{K}^*_{i,j}(\tau_1, \tau_2) := \bar{K}_{i,j}(\tau_1, \tau_2) \cdot d^{\ell}$$

The complete Kendall's tau requires a large computational cost for every single vertex in the graph. Kendall's tau takes around $O(n^2)$ where n is the number of vertices, which is a large cost to pay for every single vertex in the graph. This cost is not too prohibitive for smaller graphs such as the HITS subgraphs since they are far smaller so we can reasonable compute the full Kendall's tau with weighting on those HITS subgraphs. However, PageRank runs on the entire Web graph, and thus computing Kendall's full tau would be too expensive. Instead, we also use a variant of Kendall's tau where we only consider the first k elements from each ranking, as described in [8], with the further damping factor modification proposed above. In this variant of Kendall's tau implementation, d and k are related by equation (6):

$$d = \left(\frac{2}{n(n-1) - k(k-1)}\right)^{\frac{1}{k}} \qquad (6)$$

We can choose k based on a computational cost and then compute d accordingly. We choose d and k to be related in this way so the weighted Kendall's tau increases linearly with k instead of quadratically. This relationship also forces the contribution of all inversions outside of the top k el-

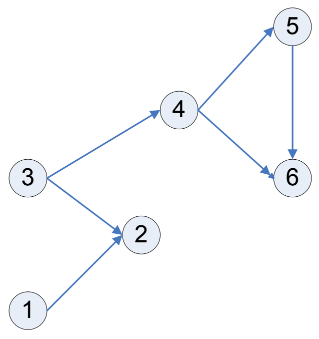

Figure 1: Non-Monotonicity

ements to contribute at most 1 to the weighted Kendall's tau.

3.4 Non-Monotonicity

In this subsection, we show a significant difference between PerturbationRank and PageRank/HITS. First of all, let us recall the definition of "monotonicity" according to [22]. For a Web page p, denote by $B(p)$ the set of Web pages that point to p.

Definition 1. *[22] A link analysis ranking algorithm is* monotone *if for every graph G, and for every pair of nodes j and k, if $B(j) \subseteq B(k)$, then $\pi(j) \leq \pi(k)$.*

Borodin et al. [22] showed (Theorem 4.29) that many known ranking algorithms, including PageRank, HITS, counting in-degree, and SALSA, satisfy this property.

Consider the 6-vertex graph in Figure 1. Node 4 is only pointed to by node 3 while Node 2 is pointed to by Node 1 and 3. Therefore, if a ranking algorithm satisfies the monotonicity, Node 2 should rank higher than Node 1. However, PerturbationRank on PageRank gives node 2 a score of 0.1927 while gives node 4 a score of 0.2133. Furthermore, PerturbationRank on HITS gives Node 2 an authority score of 0.3965 and gives Node 4 an authority score of 0.4624. Thus, both algorithms rank Node 4 higher than Node 2. Therefore, PerturbationRank, when on PageRank and HITS, is not monotone.

Although Figure 1 is a toy example, the actual Web graph could have many substructures like this. For example, Node 3 could be the home page of a computer science department; Node 4 could be the home page of a research lab in the department; and Node 4 points to Node 5 and 6, which may contain more detailed information about the research lab; Node 2 could be a Web page that has the contact information(or legal disclaimer information) of the department; Node 1 could be a Web page outside the department. If a Web user wants to find some useful information about "computer science", the user probably prefers Node 4, which contains information about a computer science research lab, to Node 2, which merely has the contact information (or legal information). In an exaggerated case, Node 4 could point to a large set of nodes besides Node 5 and 6. Those nodes pointed by Node 4 could also have complicated interconnection structure. This structure possibly implies that i) Node 4 is a home page; ii) Node 4 as well as the nodes pointed by it could be information intensive and relevant to a specific query word. Thus, a user may consider Node 4 more important than Node 2. Such notion of importance cannot be captured by a monotone ranking algorithm.

A second drawback of monotonicity is that it allows link spammers to boost the ranking of a target page easily. That is, if a link spammer wants to boost the ranking of Node 2, he can create a boosting page Node 1 and make it point to Node 2. This strategy will boost the PageRank score of the target page Node 2 [7]. Fortunately, it is no longer guaranteed to work under our PerturbationRank.

3.5 Efficiency of Implementation

To compute PerturbationRank we need to re-compute the ranking of the Web pages after each Web page is disconnected. Thus if there are M Web pages and it takes the base ranking algorithm T_M steps to rank them, computing the PerturbationRank takes $O(MT_{M-1}) \approx O(MT_M)$ steps. For PerturbationRank on top of HITS, this does not pose a complexity issue, since the computation is restricted to the subset of Web pages containing the query and some neighboring pages. However, when M is the size of the whole Web, a situation that arises if PageRank is the base algorithm, the complexity is daunting. Thus we always restrict the computation of PerturbationRank to a subset of Web pages identified by the same method used in HITS.

We point out, however, when it is desirable to deal with large M, it is possible to moderate the complexity by approximating PerturbationRank, instead of computing it exactly. We will describe an approach for balancing the efficiency and the accuracy for approximating PerturbationRank at the discussion section.

3.6 Connections with Related Works

"Betweenness centrality" [24], which was studied intensively by physicists, is another way to characterize the importance of a vertex in a network. The betweenness centrality of a vertex is the number of geodesic paths between all pairs of vertices that go through the vertex. It is easy to see that betweenness centrality is not a monotone measure if it is used as a ranking method. However, betweenness centrality may not be an effective indication for the relevance of a Web page in the context of search. For example, a geodesic path between s and t that goes through a Web page p may not be a strong indication of the relevance of p to a query, although it still contributes one vote to the betweenness score of Web page p. The reason is when page s has a large distance from p on the geodesic path, it probably implies that page p and s are not closely relevant. Thus, such a path should not contribute much to the authority score of Web page p. However, betweenness centrality may be more suitable in the situation of evaluating the robustness of a computer networks. We discuss this in more details in Section 5.

"AUTHORITYAVG" [22] is another non-monotone ranking algorithm, where the authority score of a vertex is the average of the hubs scores of the vertices that point to it. However, PerturbationRank, which measures the importance of a vertex by the amount of changes it can cause, follows fundamentally different philosophy from AUTHORITYAVG.

4. EVALUATION

4.1 Experimental Setup

4.1.1 Queries and Graphs

In this Section, we evaluate the performance of PerturbationRank and compare it with that of the base ranking algorithms. We follow the methodology of Borodin et al., use the same dataset and a subset of their queries.

> `abortion, alcohol, computational complexity, computational geometry, genetic, geometry, globalization, iraq war, movies, national parks, shakespeare, weather.`

Following [22], for each query, the query-specific Web graph is generated as follows. The first 200 Web pages returned by Google related to that query forms of the *root set*. Then the root set is expanded to the *base set* by including the pages pointing to or pointed to by the pages in root set. In particular, only the first 50 pages pointing to a page in the root set are kept in the base set. All the query-specific graphs are available online [2]. However, since the datasets were collected in 2003, some of the URL links are already dead. In order to deal with this problem, we retreived the cached pages from Archive.org[3].

4.1.2 Implementations

The four ranking algorithms we implement are the PerturbationRank on PageRank, the PerturbationRank on HITS, original PageRank and original HITS. For the PerturbationRank on PageRank, we use l_1 norm to measure the disruption and the final ranking vector is normalized under l_1 norm while for the PerturbationRank on HITS, we use l_2 norm to measure the disruption and the final ranking vector is normalized under l_2 norm. For the PageRank algorithm, we set the random jump probability to be 0.15. For PageRank and HITS, iterative methods are applied to compute the ranking vector. The iterative methods stops if the difference between two successive ranking vectors is smaller than 10^{-7} or 200 iterations are completed.

For each ranking algorithm, the top 10 ranked Web pages in the *root* set are evaluated by 5 human users. This is different from the measure used in [22], where the top 10 Web pages in the *base* set given by the ranking algorithm are evaluated. This latter method is not ideal for us, since a Web page to be evaluated could be a page in the base set but not in the root set. According to our definitions, the root set contains the first 200 Web pages returned by Google upon a specific query, which should be relevant to the query word. However, a Web page in the base set may not contain the query word at all, which is considered to be irrelevant. Hence, the measure in [22] will reduce the qualities of all the ranking algorithms significantly and artificially. For example, according to [22], HITS does not return any relevant documents at all on a couple of queries. In practice, a search engine will build up an inverted list for each query word, which includes all the Web pages that have query word as the content. Thus a Web page, which does not contain the query word, can not be in the returned Web pages list of the search engine at all. Therefore, our measure, which only considers the Web pages in root set, is a better approximation of the measure used by real search engines than the measure used in [22].

[2]http://www.cs.toronto.edu/ tsap/experiments
[3]http://www.archive.org/Web/Web.php

	$\|\pi - \pi p\|_1$	$\frac{\tau(\pi, \pi p)}{N(N-1)/2}$	$\|a - ap\|_2$	$\frac{\tau(a, ap)}{N(N-1)/2}$	No. of nodes
abortion	0.4340	0.3067	0.0039	0.0983	3340
alcohol	0.3566	0.1503	0.0923	0.0889	4594
computational complexity	0.4495	0.3115	0.2592	0.1038	1075
computational geometry	0.4213	0.3131	0.4817	0.0918	2292
genetic	0.3808	0.2024	0.2076	0.0850	5298
geometry	0.3788	0.2651	0.1999	0.0994	4326
globalization	0.4928	0.3386	0.0433	0.0878	4334
iraq war	0.4670	0.2944	0.3382	0.1363	3782
movies	0.3950	0.2125	0.7223	0.1063	7967
national parks	0.4204	0.1809	0.0104	0.0754	4757
shakespeare	0.3630	0.1666	0.2188	0.0961	4383
weather	0.3228	0.2184	0.0965	0.1272	8011
mean	0.4068	0.2467	0.2228	0.0997	4513
stdev	0.0496	0.0655	0.2111	0.0173	1990

Table 1: The Disruptions of Rankings

	PR	PPR	HITS	PH
PR	10	8	3.75	3.58
PPR	8	10	3.83	3.58
HITS	3.75	3.83	10	9.08
PH	3.58	3.58	9.08	10

Table 2: Correlations among top 10 Web pages

We focus on the top 10 Web pages because the first returned page of a search engine usually contains about 10 URL links. Given the fact that most users of a search engine are unlikley to spend time examining many returned links, the qualities of the top 10 Web pages are crucial for the overall impression of the search quality.

4.1.3 Users Evaluation

Following [22], for each Web page returned by a ranking algorithm, the human users are asked to rate it as “Highly Relevant”,“Relevant”, “Irrelevant” and “Do not know”. A user rates a Web page as “highly relevant” if he/she would like the URL of the Web page to appear in the top few returned results by the ranking algorithm; as “relevant” if he/she would like the URL of the Web page to appear in the first page of results returned by the ranking algorithm; as “irrelevant” if he/she does not think the Web page contains useful information. The rating should be “Do not Know” if thet user cannot judge the relevance of the Web page. We mixed the Web pages returned by the 4 ranking algorithms and ask 5 human users to rate them. Given each person’s rating, a Web page is rated as relevant if the relevant and highly relevant votes are more than the irrelevant and “do not know” votes; otherwise, it is rated as irrelevant. Finally, for a Web page rated as relevant, if its highly relevant votes are more than relevant votes, it is rated as highly relevant instead of relevant.

4.2 Comparisons of the Rankings

For simplicity, denote PageRank by *PR*, PerturbationRank on PageRank by *PPR*, and PerturbationRank on HITS by *PH*. Let π, π_p, a and a_p be the ranking vectors of PR, PPR, HITS (authority) and PH, respectively. We introduce the following metrics to measure the output correlations. For PR and PPR, we use the ℓ_1 distance $\|\pi - \pi_p\|_1$, as well as the

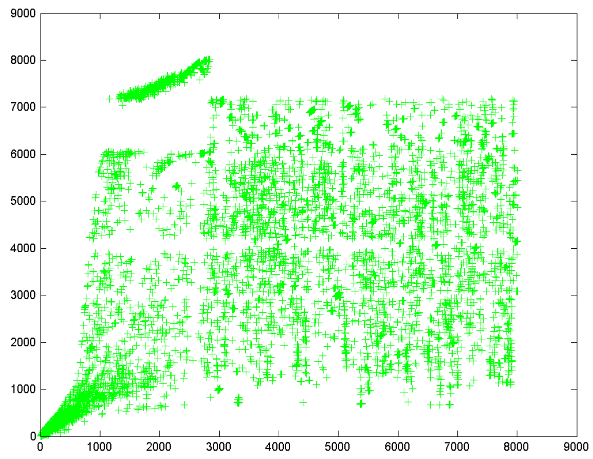

Figure 2: Correlation between PR and PPR for `weather`

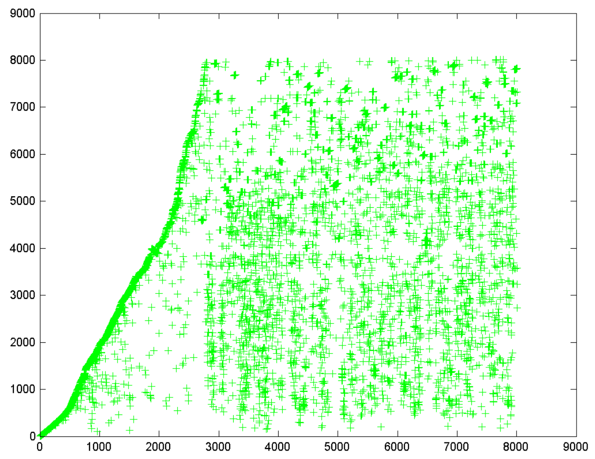

Figure 3: Correlation between HITS and PH for `weather`

normalized Kendall's tau: $\frac{\tau(\pi,\pi_p)}{N(N-1)/2}$. The maximum values they can take are 2 and 1, respectively, and the smaller they are, the higher correlation. Note that the two metrics are not equivalent. That is, one could be large while the other is small. Therefore, we consider both. For HITS and PH, we use the ℓ_2 distance $\|a - a_p\|_2$ (with the maximum value of $\sqrt{2}$), and the normalized Kendall's tau. The change from ℓ_1 to ℓ_2 is because the HITS vectors are more naturally interpreted in the ℓ_2 norm, while PageRank vectors are naturally probability distributions with a unit ℓ_1 norm.

Table 1 presents the values for the four metrics for each query, showing considerable correlations but also substantial difference between PerturbationRank and its base algorithms. In particular, Kendall's tau shows that for PR and PPR, the orders of 24.67% of pairs of pages are different while for HITS and PH, the ratio is 9.97%, which shows a higher correlation on HITS.

In computing the Kendall's tau computed in Table 1, we assume equal importance of each Web page. However, as we argued before, in practice, users may be only interested in top pages returned by a search engine. Thus, besides calculating the ranking correlations based on all the pages in a Web graph, it is important to measure the correlation between the top ranked pages, which we show in Table 2. Each entry represents the number of common pages in top ten list shared by the row and column ranking algorithms. The data show that on average, PR and PPR share 8 out of the top 10 pages, while HITS and PH share about 9. This confirms our conclusion above: PerturbationRank and base ranking algorithms are considerably correlated but substantially different, even in the top 10 list. Note that a small difference in the top 10 list is significant, since it gives the user noticeably difference experience.

Finally, we plot the ranking correlations for the query `weather`, which will give us some intuitive impressions about how PerturbationRank differs from the base ranking algorithms. In particular, we will show some correlations between PPR/PH scores and out-degrees of pages that are greatly demoted by PerturbationRank. In Figure 2, the X axis is the rank of each page given by PR while the Y axis is the rank given by PPR. First of all, the crosses ('+') roughly concentrated along the diagonal line for the top 1000 pages. This implies that highly ranked pages under PR likely have high rankings under PPR. This is because the relatively high PR scores of those top ranked Web pages under PR can contribute a significant amount to their PPR scores. However, the rankings of Web pages, which are not ranked in top 1000 under PR, seem to have weak correlations with their rankings under PPR. This is also conceivable. For Web pages with relatively low PR scores, although their own PR scores can not contribute a lot to their PPR scores, they may link to some pages with high PR scores, which can still make them important under PPR. However, what kinds of pages a Web page points to is roughly random. That is why there is weak correlation between PR rankings and PPR rankings for those pages not in the top 1000. At last, as shown in the upper left corner of Figure 2, some pages ranked between 1000 and 3000 by PR have very low rankings (more than 7000) under PPR. Note that for those pages, the average in-degree is 2.182 while the average in-degree of the whole graph is 4.328. This makes those pages have relatively high rankings under PR. However, the average out-degree of those pages is 0.025. Because of the violation of monotonicity by PerturbationRank, the removal of the links of those pages may not cause significant changes over the Web. Meanwhile, as shown in Figure 3, HITS&PH have similar behaviors as PR&PPR. In particular, for those pages ranked between 1000 and 3000 by HITS but having PH rankings lower than their HITS rankings by more than 100, the average out-degree is 1.703 while the average out-degree of the whole graph is 4.328. The relative lower out-degrees of those pages may cause them to have lower rankings under PH. This is shown as the tense green belt in Figure 3. Again, this might be the result of the violation of monotonicity.

In general, the dynamics of PerturbationRank on PageRank/HITS is complex and global. Thus, we can not accurately capture the behaviors of PerturbationRank by simply looking at the degree information of a Web page instead of actually computing it. The analysis above is relatively rough. However, we regard it as the first step to study the dynamics of PerturbationRank.

	PR	PPR	HITS	PH	PPR /PR	PH /HITS
abortion	50%	60%	20%	20%	33%	-
alcohol	20%	20%	20%	20%	0%	-
computational complexity	60%	50%	50%	50%	0%	-
computational geometry	20%	20%	30%	30%	0%	0%
genetic	10%	10%	0%	0%	0%	0%
geometry	0%	0%	0%	0%	0%	-
globalization	30%	30%	20%	20%	0%	0%
iraq war	10%	10%	0%	10%	0%	25%
movies	20%	20%	30%	30%	50%	0%
national parks	20%	20%	10%	10%	0%	0%
shakespeare	10%	0%	0%	0%	0%	-
weather	30%	20%	40%	40%	0%	-

Table 3: High Relevance Ratio

	PR	PPR	HITS	PH	PPR /PR	PH /HITS
abortion	70%	90%	70%	70%	67%	-
alcohol	90%	100%	100%	100%	100%	-
computational complexity	90%	90%	90%	90%	100%	-
computational geometry	90%	90%	90%	100%	100%	100%
genetic	60%	60%	90%	90%	100%	100%
geometry	90%	80%	100%	100%	67%	-
globalization	40%	40%	50%	60%	0%	100%
iraq war	60%	60%	70%	50%	100%	50%
movies	70%	80%	100%	90%	100%	0%
national parks	60%	50%	40%	30%	33%	0%
shakespeare	60%	60%	60%	60%	67%	-
weather	80%	60%	90%	90%	0%	-

Table 4: Relevance Ratio

4.3 Users' Evaluation Results

We now show that the difference between PerturbationRank and the base algorithms may provide some useful information. As described before, 5 human users were asked to rate the pages returned by all the four rankings. Based on the users' evaluation results, we calculate the percentage of the top 10 pages by a ranking rated "Highly Relevant" by the users, as well as the percentage rated "Relevant" or "Highly Relevant". According to our definition, the relevance ratio is always no less than the high relevance ratio. The data are presented in Table 3 and Table 4. In these two tables, *PPR/PR* denotes those pages in the top 10 of PPR but not in that of PR, and *PH/HITS* is defined similarly. If there is no pages in PPR/PR or PH/HITS, a "-" is presented.

Table 3 presents the data for the high relevance ratio. As shown in the table, the values for PR and PPR are almost identical. For some queries, e.g. `abortion`, PPR performs better while for some queries, e.g. `shakespeare`, PR performs better. PPR occasionally identify highly relevant pages not identified by PR. This is a substantial advantage since one more highly relevant page in the top 10 list will greatly improve the quality of a search engine. Similarly, PH has comparable high relevant ratio with HITS and occasionally identify new highly relevant pages. Unlike PPR/PR, PH/HITS is empty for quite a few queries, which confirms our earlier conclusion that PH/HITS is more correlated than PPR/PR.

Table 4 shows the data for the relevance ratio. Again, the values for PR and PPR are similar, and neither outperforms consistently the other. Most pages in PPR/PR, however, are likely to be relevant or highly relevant. Thus PPR is able to discover useful information missed by PR. The situation for HITS/PH is similar.

We conclude this section by a few comments on the evaluation methodology. There seems no perfect method to evaluate and compare ranking qualities. Arguably the ultimate judgement is the end users' experience. Thus an ideal setup is to use an actual search engine, present to the users different rankings and use the click statistics to evaluate the performance. Unfortunately we do not have the access to an actual search engine, which makes the evaluation much more challenging.

We adopt the human evaluation methodology of Borodin et al. [22]. We point out that this method has some inherent limitations due to the ambiguity and subjectiveness in human judgement. First, different users may interpret a query phrase with a different query intention. This may contribute to, but may not be the only reason for, the different perception of a Web page. The definitions of "relevance" and "high relevance" may also be interpreted differently. Thus, a page rated as "highly relevant" by one user may be rated as "relevant" or even (although less likely) "irrelevant" by another user.

Other than the human factor, a second source of noise comes from the fact that in reality, the overall performance of a ranking system depends not only on the ranking algorithm itself, but also on the subset of Web pages retrieved from the query and used as the input for the algorithm. For example, Google uses the inverted list to index the set of pages relevant to a query while here we use the root set in a query-specific graph as an approximation. An additional issue for our experiment is that the content of some of the Web pages in our data set have changed over time. This may cause some pages to be rated as "irrelevant". With those issues in mind, we need to be cautious in interpreting our experimental results. We thus only draw qualitative conclusions, other than quantitative ones.

5. DISCUSSIONS

5.1 Computational Efficiency

Although PerturbationRank on PageRank is run on query-specific graphs in our experiments, which does not impose heavy computational cost, it is still worthwhile to discuss the computational efficiency problems in more depth. As noted in Section 3.5, when the size of the Web pages that PerturbationRank works on is large, the computational complexity may pose a challenge. We sketch below an approach for moderating the complexity through approximation. We draw our inspirations from the many works on the following problem: "How to update the PageRank vector without recomputing it from scratch, say, using the power method?" This is an important practical problem as the Web changes frequently. PageRank is nothing but the stationary distribution of a Markov chain, and the problem of updating the stationary distribution when the Markov chain changes has a longer history than PageRank. For example, several formulas are known for computing the exact update (E.g. [18, 11]) based on the classical Sherman-Morrison formula for inverting a matrix perturbed by a rank-one matrix. However, those updating procedures become inefficient when multiple columns are changed in the transition matrix. Thus researchers have developed algorithms for approximate updates that are efficient and accurate [6, 16, 12].

Our problem is more complicated as we need to compute M updates. To simplify the computation, we start with the hypothesis most of the change induced by removing the incoming and outgoing links of a Web page v takes place in a certain neighborhood $N(v)$ of v. For example, $N(v)$ could be the set of Web pages having a link from or to v, or it could be the set of Web pages for which the mean hitting time to v is smaller than some threshold. There are techniques (e.g. in [6]) to "compress" the huge transition matrix for the whole Web to a new chain on $|N(v)| + 1$ states, one for each element in $N(v)$, and the last one for the rest of the surviving Web pages combined. We can then compute the stationary distribution of this compressed chain, and use it to compute the ℓ_1 difference of the perturbed and the original stationary distributions in time $O(|N(v)|)$. Thus all the M updates can be computed in time $O(Mpoly(|N(v)|)$. The choice of $N(v)$ is also critical for the accuracy.

5.2 Potential Applications of PerturbationRank

We now briefly discuss other potential applications of PerturbationRank.

Combating spamdexing. Improving a Web page's ranking according to a popular search engine means much more traffic to the site and translates to higher profit for online businesses. Thus many techniques have been used to manipulate links or contents of Web pages solely to improve the ranking of the Web page in a targeted search engine. Those techniques, referred to as *spamdexing*, do not in general improve the importance of the Web page from a user's perspective, thus have reduced the effectiveness of search engines. It appears that spamdexing is a serious problem — a significant percentage of Web pages indexed by search engines are spam [10, 9, 4]. This undesirable situation motivates many studies on how to combat spamdexing [10, 9, 4, 7, 17, 21].

PerturbationRank may make link-based spamdexing much more difficult. This is because, intuitively, it is difficult for a spammer to substantially increase the dependence of the *whole* Web on the Web page whose ranking is to be boosted. Thus, if the measurement of disruption gives high weights to changes that are less likely caused by spam, the corresponding PerturbationRank may be hard to manipulate. For example, a good measurement of disruption for this purpose could assign higher weights to Web pages further away from the target page, or one could measure the perturbation on a set of "trusted" pages. Defining and evaluating an efficiently computable PerturbationRank resilient to spamdexing is a promising direction.

The security and robustness of networked systems. PerturbationRank can be applied to different levels of a layered networked system. For example, a vertex on the graph where we apply PerturbationRank could be an individual server, a cluster of servers sharing a physical link, or a domain, etc. Then the removal of the links of the vertex corresponds to the failure of the Web server, a cluster, or a domain. Since PerturbationRank measures the disruption once the corresponding entity is not available, it also evaluates the security risk to which the failure of the entity exposes the whole system. Thus PerturbationRank can be used to guide the allocation of resources for safe-guarding or maintaining the robustness the system. To be more specific, consider the following hypothetical scenario. A Webmaster, who knows traffic pattern between her servers may use PerturbationRank to estimate the disruption to the overall traffic of her site if a server is down. Based on the PerturbationRank, she may decide how many mirrors each server should have.

6. CONCLUSIONS

We have proposed PerturbationRank, a new approach for ranking Web pages. In this approach, the importance score of a Web page is a quantity measuring the amount of change on the whole Web resulting from disconnecting the Web page. In particular, we focus on the following method for quantifying the change: use the output of a known ranking algorithm to represent the state of the Web, and apply an appropriate distance metric on the two ranking outputs before and after the disconnection. The new ranking algorithm, PerturbationRank, is fundamentally different from traditional ranking algorithms, such as PageRank and HITS, as it violates monotonicity. Following the method of Borodin et al. [22], we evaluate PerturbationRank using a real data set and human ratings of returned Web pages. Our experiments demonstrate that the new and the base rankings are considerably correlated, of comparable performances, and at the same time substantially different. We conclude that PerturbationRank gives an alternative and useful ranking of Web pages.

7. REFERENCES

[1] A. Altman and M. Tennenholtz. Ranking systems: The pagerank axioms, 2004.

[2] K. Arrow. *Social Choice and Individual Values*. Yale University Press, New Haven, Connecticut, second edition, 1963.

[3] R. Baeza-Yates, P. Boldi, and C. Castillo. Generalizing pagerank: Damping functions for link-based ranking algorithms. In *Proceedings of ACM SIGIR*, pages 308–315, Seattle, Washington, USA, August 2006. ACM Press.

[4] A. A. Benczúr, K. Csalogány, T. Sarlós, and M. Uher. Spamrank–fully automatic link spam detection. In *AIRWeb*, pages 25–38, 2005.

[5] K. Bryan and T. Leise. The $25,000,000,000 eigenvector: The linear algebra behind google, 2006.

[6] S. Chien, C. Dwork, R. Kumar, D. R. Simon, and D. Sivakumar. Link evolution: analysis and algorithms. *Internet Math.*, 1(3):277–304, 2004.

[7] Y. Du, Y. Shi, and X. Zhao. Using spam farm to boost pagerank. In *Proceeding of Third International Workshop on Adversarial Information Retrieval on the Web*. ACM, 2007.

[8] R. Fagin, R. Kumar, and D. Sivakumar. Comparing top k lists, 2003.

[9] D. Fetterly, M. Manasse, and M. Najork. Spam, damn spam, and statistics: Using statistical analysis to locate spam Web pages. In *Proceedings of the 7th International Workshop on the Web and Databases*, pages 1–6, Paris, France, June 2004.

[10] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating Web spam with trustrank. In M. A. Nascimento, M. T. Özsu, D. Kossmann, R. J. Miller, J. A. Blakeley, and K. B. Schiefer, editors, *VLDB*, pages 576–587. Morgan Kaufmann, 2004.

[11] J. J. Hunter. Stationary distributions of perturbed Markov chains. *Linear Algebra Appl.*, 82:201–214, 1986.

[12] I. C. F. Ipsen and S. Kirkland. Convergence analysis of a PageRank updating algorithm by Langville and Meyer. *SIAM J. Matrix Anal. Appl.*, 27(4):952–967 (electronic), 2006.

[13] J. G. Kemeny and J. L. Snell. *Finite Markov Chains*. Van Nostrand Reinhold, New York, 1960.

[14] M. Kendall and J. Gibbons. *Rank Correlation Methods*. Edward Arnold, London, United Kingdom, 1990.

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

[16] A. N. Langville and C. D. Meyer. Updating Markov chains with an eye on Google's PageRank. *SIAM J. Matrix Anal. Appl.*, 27(4):968–987 (electronic), 2006.

[17] F. Luccio and L. Pagli. Web marshals fighting curly link farms. In P. Crescenzi, G. Prencipe, and G. Pucci, editors, *FUN*, volume 4475 of *Lecture Notes in Computer Science*, pages 240–248. Springer, 2007.

[18] C. D. Meyer, Jr. and J. M. Shoaf. Updating finite Markov chains by using techniques of group matrix inversion. *J. Statist. Comput. Simulation*, 11(3-4):163–181, 1980.

[19] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

[20] Y. Wang and D. J. DeWitt. Computing pagerank in a distributed internet search system. In *vldb'2004: Proceedings of the Thirtieth international conference on Very large data bases*, pages 420–431. VLDB Endowment, 2004.

[21] B. Wu and K. Chellapilla. Extracting link spam using biased random walks from spam seed sets. In *AIRWeb '07: Proceedings of the 3rd international workshop on Adversarial information retrieval on the web*, pages 37–44, New York, NY, USA, 2007. ACM.

[22] Allan Borodin, Gareth O. Roberts, Jeffrey S. Rosenthal and Panayiotis Tsaparas. Link analysis ranking: algorithms,theory, and experiments. In *ACM Trans. Interet Technol.*, 5, 1, 2005, pages 231–297, ACM, New York, NY, USA.

[23] Langville, Amy N. and Meyer, Carl D., Google's PageRank and Beyond: The Science of Search Engine Rankings. Princeton University Press, 2006.

[24] M. E. J. Newman, The structure and function of complex networks, SIAM Review, 2003, 45, 167–256.