

Rank Computation Methods for Web Documents

[Maxim Lifantsev](#)

[Department of Computer Science, SUNY at Stony Brook,
Stony Brook, NY 11794, U.S.A.](#)

maxim@cs.sunysb.edu

Abstract

We analyze the existing rank computation methods for the Web using a notion of rank flows and focusing on the methods for authority rank computation. We then propose a new model, called the voting model, for computing various ranks of Web pages that can be used by a search engine to better estimate the quality and relevance of the Web pages. The proposed model has a number of advantages over the surfing model used in the Google search engine [4]: it subsumes the surfing model providing natural trivially converging computation of more different types of ranks, as well as easier extensibility, faster personalization, and finer control over its parameters, due to its conceptual simplicity. The voting model also provides a new meaning for the parameters and the results of the Google's surfing model.

Keywords: Search Engines, Web links, Metadata, Ranking Web Pages, Google.

1. Introduction

It is well known that the Web poses a challenge for locating quality information: the ease of publishing on the Web leads to enormous diversity of the accessible information in topic, quality, presentation, style, etc., as well as astonishing growth of available data. Traditional information retrieval techniques employed by most major search engines rely almost solely on the text of the Web pages viewed separately and are notorious for their inability to adequately rate the relevance and quality of all the numerous pages that are potentially relevant to the user's query [6]. Thus, new methods to improve the ability of users to locate relevant information on high quality Web documents become increasingly important.

Innovative information retrieval methods such as those used by the Google search engine [1,4] and the Clever project [2] try to use structural information present in HTML documents and more importantly the non-local link structure of the Web, thus leveraging the distributed knowledge of many Web page authors reflected by the link structure. The main idea of link-based approaches is that links generally signify approval of the linked document and its relevance to the topic of the linking document. It is therefore possible to compute some approval, popularity, or quality ranks of a page based on Web connectivity. The link structure analysis employed by Google and Clever

mechanically produces very impressive quality/relevance/importance judgments compared to the traditional search engines [4,3].

In this paper we first discuss different link-based rank computation methods proposed in the literature and analyze them using the universal notion of **rank flows**. We then concentrate on methods to determine authority ranks: First we review the PageRank method [10] used by Google [4] by focusing on the underlying model (called the **surfing model** here) and the meaning of its parameters and results. Next we propose a new authority rank computation model: the **voting model**. The new model possesses a number of advantages over the surfing model used in the PageRank method: It is a bit more natural and easier to understand. It subsumes the surfing model, thus allowing us to analyze the latter from a different viewpoint. It is more flexible, extensible, and personalizable. In particular we present its extensions to support generation of jury-style absolute scores with confidence and to support links representing disapproval (only a subcase of the latter extension can be retranslated back into the surfing model). The new model also allows for easy collection of many different types of rank statistics through trivially converging computation.

2. Rank Flow Model

A common theme of the link-based information retrieval methods for the Web is that they compute some ranks of Web pages that reflect quality, relevance, or importance of a page. The ranks are computed following some flow model: the ranks flow along the links among the documents.

2.1 Types of Rank Flows

We can distinguish the following types of rank flows:

Authority rank flow

It flows forward on links: if page A points to page B , then mostly that means that the author of A thinks that B is relevant to (or good on the topic of) A , hence A gives some of its authority to B .

These ranks are about what pages are of high quality or relevance or are popular. This is also about how frequently one reaches (is recommended) a given page when exploring the Web for the desired information [10]. This kind of flow is used in the Google search engine [1]. ¹

Resource rank flow

It flows backward on links: if page A points to page B , then A contains the information from page B to some extent because B can be easily reached from A . ²

These ranks are about how easy it is to reach good relevant information starting from a given page. This flow was proposed in [7]. Its variation that stops after following one link

is used in the Clever project [5] to determine what are the good hub pages. 3 Note that for this flow we need some precomputed measure of relevance (e.g. authority ranks) to start with.

Authority determination rank flow

It also flows backward on links: if page **A** points to page **B** thus giving a certain amount of authority rank to **B**, then **A** has the deciding power over **B**'s rank to the corresponding extent. This backward authority determination flow allows one to determine the sites whose word (links) contributes most to the ranks of pages we started the flow from. Note that here we again need the authority ranks to start with.

These ranks are about how influential a given page really is in the community, whereas authority ranks are about pages' potential degree of influence. 4

2.2 Related Work

The two major link-based rank computation algorithms that have been successfully applied in practice are the PageRank method of Google [10,1] and different variations of the Clever algorithm (initially known as HITS) [5,2,3]. The PageRank method (see section 3) is used to compute authority-like ranks according to its surfing model for all the documents a search engine has crawled. These ranks are independent of the queries, i.e. a page gets high rank if linked from many high-ranked pages. The Clever algorithm computes authority and resource ranks using an iterative method that relies on the current estimate of authority ranks for the next estimate of resource ranks and vice versa. Pages get high authority ranks if linked from many pages with high resource ranks, which are called **hubs**. This algorithm considers just a small set of Web pages that are relevant to the query and obtained using a traditional search engine.

We argue below that the special iterative treatment of hub pages to generate authority ranks is not needed when authority ranks are computed using most of the relevant Web pages that exist. Therefore authority ranks can be computed by themselves without relying on estimates of other types of ranks, and the ranks of other types can then be computed from fixed authority ranks.

Good hubs are pages linking to many good authorities [5]. If we compare Web documents to research paper publications, hubs can be regarded as overview papers that in the paper-publishing case do not usually get published because they carry too little "content" as perceived by the reviewers. More generally, really good hubs correspond to comprehensive overview papers, journals, conference volumes, and libraries, but journals and conferences usually have their name established through long hard work, not just right away by lumping known good papers in the first issue, which is what needs to be done to create a page satisfying the definition of a good hub. These arguments suggest that hubs should be considered as just regular authority sources for authority ranks that are no different than other pages. That is, if a hub is really good and popular it will get its authority rank boost from the many links pointing to it.

The reason hubs are used in the Clever algorithm appears to be that it operates on a very small sample of the Web, in which case the hubs would not get the authority rank boost they usually deserve because the many pages (transitively) pointing to them are not usually present in the set

of considered pages. Hence, to get better results, hub importance has to be artificially boosted in a way that is sufficiently consistent with the authority ranks they would get if we had considered a bigger subset of the Web. Another argument against hubs is that they are easily reproducible by a single person, hence opening the door for rank manipulations. [5](#)

Although the list of pages with the highest authority ranks presumably *is* the best hub (which then implies that we do not need hubs at all), it might not exactly be the best one because it is mechanically compiled, and not manually constructed with a coherent vision. Hence, a search engine should list good hubs in search results, as this way it *directs traffic* to good manually-composed resource compilations and encourages people to create more good resource collections, which improves Web connectivity and hence the quality of the results of link connectivity analysis. [6](#) Also some hub-like ranks might be valuable for classification as they can be used to provide co-citation statistics.

Acknowledging the importance of authority ranks, in the rest of the paper we mainly discuss the methods to determine authority ranks of a **set of nodes** N linked to each other when the links can represent different kinds of opinions of a node about other nodes. A **node** n for simplicity represents the largest unit that can be ranked: an independent Web page authoring entity. We mostly ignore the problems of determining the boundaries of such entities, taking into account their composite representation by a set of pages, and representing or extracting different opinions we associate with links.

3. Surfing Model

In this model there are surfers that go from one node to another using the links and the more surfers a node "has", the higher its authority rank. This model was first proposed in [\[10\]](#), where it is called the **random surfer model**.

3.1 Description

Initially there is a certain number of surfers (we use $|N|$ surfers [7](#) that can split themselves into fractional surfers to follow different paths). They are initially distributed evenly among all the nodes if we assume that all nodes are a priori equally popular. We represent these initial surfers by the column **start surfers vector** $S = (s_n)$ such that $\sum_{n \in N} s_n = |N|$, $s_n \in [0, |N|]$ for $n \in N$, and below we assume $s_n = 1$ for $n \in N$ if not stated otherwise.

The surfers then travel along the links dividing themselves according to propagation fractions associated with the links. We represent these propagation fractions by **surfer propagation matrix** $P = (p_{nn'})$ such that $p_{nn'} \in [0, 1]$ is the fraction of surfers directed from node n' to

node n and $\sum_{n' \in N} p_{n n'} \in [0, 1]$ for any $n' \in N$, that is, for each node the propagation fractions of its out-links form (an incomplete [8](#)) probability distribution. We will call such matrices (**incomplete**) **column distribution matrices** (column distribution matrices are known as **column stochastic** matrices in the literature). [9](#) Note that it is possible that a node directs its surfers back to itself, and that it directs its surfers equally or not-equally along all or its out-links. (The original PageRank method [[10](#)] divides all the surfers equally among all the outgoing links.)

3.2 Computation

We can compute the authority ranks (represented by the column **rank vector** \mathbf{R}) as the number of surfers located in a node after (infinitely) many propagations starting from the initial configuration: [10](#)

$$\mathbf{R} = \mathbf{P} \cdot \dots \cdot \mathbf{P} \cdot \mathbf{S}$$

The problems with this computation are that first, if there are out-links from a group of nodes and no in-links then all the surfers will eventually leave these nodes. Second, if there are in-links to a group of nodes and no out-links then all the surfers entering these nodes will not be able to leave this node group. They also might eventually disappear if some of these nodes do not propagate all the surfers they have.

To compensate for the loss of surfers due to unpropagated surfers one must either add new surfers into the system before each propagation or require that \mathbf{P} is column stochastic. To solve the above surfer leakage and capturing problems one can down-scale the propagated surfers by an **exploration probability** ϵ and add $1 - \epsilon$ times the current number of surfers back into the system (using \mathbf{S} as the distribution) to compensate for the loss of surfers caused by the down-scaling.

Hence, if \mathbf{P} is a column distribution matrix, \mathbf{S} is (element-wise) positive, and $\epsilon \in (0, 1)$, then we can compute the ranks iteratively until we reach a fixpoint using the following formulas:

$$\begin{aligned} \mathbf{R}_0 &= \mathbf{S} \\ \mathbf{R}_{i+1} &= \epsilon \mathbf{P} \cdot \mathbf{R}_i + (1 - \epsilon) \mathbf{S} \end{aligned}$$

In fact, we always reach a fixpoint such that \mathbf{R}_∞ is the maximal eigenvector of

$\epsilon \mathbf{P} + \frac{(1-\epsilon)}{|N|} \mathbf{S} \cdot \mathbf{1}^T$ that corresponds to the maximal eigenvalue of $\mathbf{1}$, where $\mathbf{1}$ is the

vector consisting of all ones: We can rewrite $\epsilon \mathbf{P} \cdot \mathbf{R}_i + (1 - \epsilon) \mathbf{S}$ as

$(\epsilon \mathbf{P} + \frac{(1-\epsilon)}{|N|} \mathbf{S} \cdot \mathbf{1}^T) \cdot \mathbf{R}_i$ when $\|\mathbf{R}_i\|_1 = |N|$ for all i . [11](#) Because

$\epsilon \mathbf{P} + \frac{(1-\epsilon)}{|N|} \mathbf{S} \cdot \mathbf{1}^T$ is a positive matrix, it is primitive and irreducible and the above iterative method, which is a variant of the power method for computing eigenvectors, computes the unique (up to a multiple) maximal eigenvector that is positive and corresponds to the maximal eigenvalue that is positive. In addition, because $\epsilon \mathbf{P} + \frac{(1-\epsilon)}{|N|} \mathbf{S} \cdot \mathbf{1}^T$ is column stochastic ($\frac{1}{|N|} \mathbf{S} \cdot \mathbf{1}^T$ is), the maximal eigenvalue is $\mathbf{1}$ (these are all standard results, see e.g. [\[9\]](#)).

When \mathbf{P} is an incomplete column distribution matrix, we can compute the ranks as the principal eigenvector of the same matrix as above, except that its eigenvalue is going to be less than $\mathbf{1}$ requiring us to normalize-up \mathbf{R}_i at each iteration step, so that we can prevent the fixpoint \mathbf{R}_∞ from becoming a zero vector. The algorithm in this case is

$$\begin{aligned} \mathbf{R}_0 &= \mathbf{S} \\ \mathbf{R}_{i+1} &= \alpha_i (\epsilon \mathbf{P} \cdot \mathbf{R}_i + (1 - \epsilon) \mathbf{S}) \end{aligned} \tag{1}$$

where α_i is such as to make $\|\mathbf{R}_{i+1}\|_1 = |N|$.

A similar, but slightly different result can be obtained by the following modified method (it was also proposed in [\[10\]](#)):

$$\begin{aligned} \mathbf{R}_0 &= \mathbf{S} \\ \mathbf{R}_{i+1} &= \epsilon \mathbf{P} \cdot \mathbf{R}_i + \alpha_i \mathbf{S} \end{aligned}$$

where α_i is such as to make $\|\mathbf{R}_{i+1}\|_1 = |N|$. The difference here is that we compensate for

the loss of unpropagated surfers by adding surfers proportionally to \mathbf{S} , rather than by consistently increasing the number of surfers in each node by the same factor. This computation finds (when it converges) the eigenvector of $\epsilon \mathbf{P} + \frac{\alpha}{|N|} \mathbf{S} \cdot \mathbf{1}^T$, where α is such that the

eigenvector has the eigenvalue of $\mathbf{1}$. [12](#) But the unmodified method ([1](#)) seems to be better because of the cleaner eigenvector semantics and properties and the fact that it has a more natural

rank loss compensation: it increases the current ranks proportionally rather than by addition.

3.3 Analysis

In the case of the unmodified method (1), the exploration probability ϵ (called damping factor in [1]) is the probability of a surfer continuing using the links in the current node to get to the new one instead of choosing a new starting node according to distribution $\frac{1}{|N|} \mathbf{S}$. Hence, ϵ

regulates the relative importance of \mathbf{S} versus \mathbf{P} , it also determines the distribution of the number of links traversed by a surfer before it decides to start over from a random node according to \mathbf{S} .

Under any of the above rank computation methods $\frac{1}{|N|} \mathbf{R}_{\infty}$ has a definite meaning: it is the probability distribution of a random surfer being at the corresponding node at the beginning of each iteration after infinitely many surfing iterations. But the value of $\mathbf{R} = \mathbf{P} \cdot \mathbf{R}_{\infty}$ instead of the fixpoint \mathbf{R}_{∞} itself seems to be the best reflection of the authority ranks, which are supposed to reflect how good each node is:

- When \mathbf{P} is an incomplete column distribution matrix, $\|\mathbf{P} \cdot \mathbf{R}_{\infty}\|_1$ will be accordingly smaller than $|N|$.

- $\mathbf{P} \cdot \mathbf{R}_{\infty}$ in a sense is less polluted by the technicalities such as ϵ , $(1 - \epsilon) \mathbf{S}$, and α_i that we have used to compute the fixpoint: if we have two nodes and a column distribution

matrix $\mathbf{P} = \begin{pmatrix} 0.1 & 0 \\ 0.9 & 1 \end{pmatrix}$, then for $\epsilon = 0.5$ we get that $\mathbf{R}_{\infty} = \begin{pmatrix} 0.52 \\ 1.47 \end{pmatrix}$ and

$\mathbf{P} \cdot \mathbf{R}_{\infty} = \begin{pmatrix} 0.052 \\ 1.947 \end{pmatrix}$. The rank in $\mathbf{P} \cdot \mathbf{R}_{\infty}$ for the first node is more natural

because according to \mathbf{P} the first node directs to itself only 10% of the surfer it initially has, thus its rank must definitely be less than 0.1.

- In section 5 below we will give another well-defined meaning for \mathbf{R}_{∞} and $\mathbf{P} \cdot \mathbf{R}_{\infty}$ showing that they both correspond to valuable characteristics of the nodes, though \mathbf{R}_{∞} is an approximation of a more precise characteristic.

4. Voting Model

In this model nodes have a certain number of votes at their disposal. A node can distribute the votes it has among other nodes (including itself) either by assigning a portion of the votes to a destination node, or by giving a portion of the votes to a destination node for the latter to decide how to distribute it. The authority rank of a node is then the number of votes eventually assigned to it. The votes we use can be fractional. This model is a generalization of direct and representative voting system: a node can use its votes directly or can give them to a representative to manage on behalf of the node; any representative in turn can also use all the votes it gets either directly or indirectly by giving them to another representative.

4.1 Description

Initially each node has a certain number of votes: We use the total number of $|N|$ votes. We

represent these initial votes by the column **initial votes vector** $\mathbf{G} = (g_n)$ such that

$$\sum_{n \in N} g_n = |N|, \quad g_n \in [0, |N|] \text{ for } n \in N, \text{ and below we assume } g_n = 1 \text{ for}$$

$n \in N$ if not stated otherwise, that is we usually assume that all the nodes are equal a priori.

The votes a node has are then divided and transmitted along the links between the nodes with the result of being either assigned to a node or trusted to a node to manage. ¹³ We represent these vote propagations by the **vote assignment matrix** $\mathbf{A} = (a_{nn'})$ and the **vote trust matrix**

$\mathbf{T} = (t_{nn'})$ such that $a_{nn'} \in [0, 1]$ is the fraction of votes assigned to node n by node

n' , $t_{nn'} \in [0, 1]$ is the fraction of votes trusted to n by n' , and

$$\sum_{n \in N} (a_{nn'} + t_{nn'}) \in [0, 1] \text{ for any } n' \in N, \text{ that is, for each node the trust and}$$

assignment fractions on its out-links form (an incomplete) probability distribution. We will call these two matrices (**incomplete**) **vote distribution matrices** in these respective cases. Incomplete matrices correspond to the cases when some nodes waste some fraction of the votes they are given. ¹⁴ Note that it is possible (and natural) that a node assigns some votes it has to itself, and it is also possible (but not natural) that a node trusts some votes it has back to itself.

Note that the elements of the trust and assignment matrices need not be explicitly specified by the nodes (though they sure can be): for example in section [5](#) we will show what values can be given to these matrices using just the existing links as the input to exactly emulate the surfing model for example.

4.2 Computation

We can compute the total votes eventually assigned to the nodes (represented by the the column **vote vector** \mathbf{V}) in the following way:

$$\begin{aligned}\mathbf{V}_0 &= \mathbf{0} \\ \mathbf{U}_0 &= \mathbf{G} \\ \mathbf{V}_{i+1} &= \mathbf{V}_i + \mathbf{A} \cdot \mathbf{U}_i \\ \mathbf{U}_{i+1} &= \mathbf{T} \cdot \mathbf{U}_i\end{aligned}$$

where $\mathbf{0}$ is the column vector of all zeroes and \mathbf{U}_i is the vector representing the number of votes given to each node to manage on the i 'th stage of the computation. The vote vector \mathbf{V} is then equal to \mathbf{V}_∞ .

One possible problem with this computation is that if there is a loop in the graph induced by \mathbf{T} with all the edges marked with $\mathbf{1}$, then we will not be able to reach a state when $\mathbf{U}_\infty = \mathbf{0}$ (which is what happens otherwise). To remedy this we can introduce a **vote transfer degradation factor** d that is slightly less than one and use $\mathbf{U}_{i+1} = d \mathbf{T} \cdot \mathbf{U}_i$ instead of

$\mathbf{U}_{i+1} = \mathbf{T} \cdot \mathbf{U}_i$ (such degradation factor is also good to have if we want to encourage nodes to assign the votes directly rather than entrust them). Or we can stop the computation when \mathbf{U}_{i+1} gets close enough to \mathbf{U}_i (in this case this cyclicly trusted votes are wasted). Or we can

also require that \mathbf{T} should not have elements that are equal to $\mathbf{1}$ (which is a natural thing to do: it is reasonable that a node does not trust all its votes to just one other node, it can assigns at least some fraction of votes to itself).

Note that, unlike the computations used in the surfing model, this computation scheme, as well as its extensions, has very easy to ensure and verify convergence conditions. The reason is that we just transfer and assign the initial votes according to \mathbf{T} and \mathbf{A} until the not yet assigned votes \mathbf{U}_i get close enough to $\mathbf{0}$, whereas in the surfing model all the surfers get reshuffled at each iteration and we have to reach a fixpoint.

4.3 Simple Vote Statistics

In the voting model one can also compute various vote statistics other than the authority ranks: An

important statistic is the number of votes a node has managed (by trusting, assigning, or wasting them), we represent this by the **managed vote vector** \mathbf{M} . Then the elements of $\mathbf{M} - \mathbf{G}$ give the total numbers of votes trusted by others to the corresponding node. In the rest of this section we see how to compute \mathbf{M} (computing other statistics on the way).

An approximation of \mathbf{M} is given by the **repeated managed vote vector** $\mathbf{C} = \sum_{i=0}^{\infty} \mathbf{U}_i$. It is always element-wise greater or equal to \mathbf{M} because it may count some votes more than once if there are trust loops in the graph induced by \mathbf{T} .

$\mathbf{D}_{\%} = \mathbf{A}^T \cdot \mathbf{1}$ is the **directly assigned vote percent vector**: its elements are the percents of votes a node has that are directly assigned by it. Similarly, $\mathbf{W}_{\%} = \mathbf{1} - (\mathbf{T} + \mathbf{A})^T \cdot \mathbf{1}$ is the **directly wasted vote percent vector**: its elements are the percents of votes a node has that are directly wasted by it.

$\mathbf{D} = \mathbf{D}_{\%} \odot \mathbf{C} = (\mathbf{A}^T \cdot \mathbf{1}) \odot (\sum_{i=0}^{\infty} \mathbf{U}_i)$ is the **directly assigned vote vector**: its elements give the total number of votes directly assigned by the corresponding node. ¹⁵ Similarly, $\mathbf{W} = \mathbf{W}_{\%} \odot \mathbf{C}$ is the **directly wasted vote vector**. Note that $\|\mathbf{D}\|_1 = |\mathbf{N}|$ and $\|\mathbf{W}\|_1 = 0$ when \mathbf{A} and \mathbf{T} are vote distribution matrices (that is, when $(\mathbf{T} + \mathbf{A})^T \cdot \mathbf{1} = \mathbf{1}$).

Let us introduce two sequences of matrices: the i -**step uncycled trustedness matrix** \mathbf{E}_i and the $\leq i$ -**step uncycled trustedness matrix** $\mathbf{E}_{\leq i}$: they are defined by

$$\begin{aligned} \mathbf{E}_0 &= \mathbf{I} \\ \mathbf{E}_{i+1} &= (\mathbf{E}_i \cdot \mathbf{T}^T)^{\oplus} \\ \mathbf{E}_{\leq i} &= \sum_{k=0}^i \mathbf{E}_k \end{aligned}$$

where \mathbf{I} is the identity matrix (has zeros everywhere except the diagonal of ones) and \mathbf{B}^{\oplus} is \mathbf{B} with all the diagonal elements set to zero. The n, n' element of \mathbf{E}_i (respectively $\mathbf{E}_{\leq i}$) is

the fraction of votes trusted without cycles [16](#) along all paths of length exactly i (respectively at most i) from the node n to the node n' .

These trustedness matrices can be used to get the following statistics:

- The i -step assigned vote percent vector $D_{\%i} = E_{\leq i} \cdot D_{\%}$
- The i -step assigned vote vector $D_i = D_{\%i} \odot C = (E_{\leq i} \cdot D_{\%}) \odot C$

These are the numbers of votes such that the respective node had them at some moment and they got assigned after being first managed by that node after following up to i re-trusting steps.

- The i -step wasted vote percent vector $W_{\%i} = E_{\leq i} \cdot W_{\%}$
- The i -step wasted vote vector $W_i = W_{\%i} \odot C = (E_{\leq i} \cdot W_{\%}) \odot C$

These are the same kind of votes as D_i , except that the votes eventually got wasted instead of being assigned.

- The i -step managed vote vector
 $M_i = D_i + W_i = (E_{\leq i} \cdot (D_{\%} + W_{\%})) \odot C$
- The total managed vote vector
 $M = M_{\infty} = D_{\infty} + W_{\infty} = (E_{\leq \infty} \cdot (D_{\%} + W_{\%})) \odot C$

Note that $D_{\%} = D_{\%0}$, $D = D_0$, $W = W_0$, and $W_{\%} = W_{\%0}$.

Hence, the vector $E_{\leq \infty} \cdot (D_{\%} + W_{\%})$ gives the node-wise coefficients to down-scale the repeated managed vote vector C in order to compensate for the repeated counting due to the trust loops induced by T .

The problem with computing the above statistics lies in evaluating E_i for large values of i : if there are many long paths induced by T then each E_{i+1} can potentially contain twice as many non-zero elements as E_i . This can cause the matrices we have to work with to become non-sparse, hence it will be very expensive to do the computations when $|N|$ is very large. A

possible solution is to make some elements in \mathbf{E}_i that are close to zero into zeroes at each iteration. This will result in inaccuracies in the final results we compute that depend on which and how large non-zero elements we turn into zeroes. The cheapest way when we need just the value of $\mathbf{E}_{<\infty} \cdot (\mathbf{D}\% + \mathbf{W}\%)$ is to simply approximate it by $\mathbf{1}$.

Summarizing, we can get such valuable statistics in the voting model as

- The vote vector \mathbf{V}_{∞} , which shows how many votes each node eventually got and reflects how good the node is.
- The **trusted vote vector** $\mathbf{M} - \mathbf{G}$, which shows how many votes each node was trusted by others and reflects how influential the decisions of this node are for the vote distribution. [17](#)
- The managed vote vectors $\mathbf{M}_0, \mathbf{M}_1, \dots$, which show how many votes were assigned (or wasted) by the respective node directly, directly and indirectly by the immediately trusted nodes, and so on. They reflect (especially when compared to \mathbf{M}) how active the node is in determining the final destination of the votes it is trusted with. [18](#)
- The **wasted vote vector** \mathbf{W}_{∞} , which shows how many votes were wasted as the result of the decisions made by the respective node and in a sense reflects how indifferent the node is to the voting process (different \mathbf{W}_i 's can show finer details).

4.4 Extended Vote Statistics

An interesting extension is to compute for each node some measure of vote diversity by looking up the originators of the votes and seeing how diverse they are, as well as by looking at the paths the votes have traveled (it is also interesting to know the exact number of all the originators, as well as that of the intermediate nodes).

Another interesting statistic is to see what kind of nodes (immediately) directed the votes to a given node (with high vote value): e.g. are they many nodes with small managed vote values or are they a few nodes with high managed vote values. This will give a measure of how directly or indirectly the nodes have contributed their votes to the given node.

Note that many of the discussed statistics reflect very different yet important characteristics of a node and they can be used separately or combined using some (task-specific) statistic aggregation function to yield a compound rank value.

5. Connection Between the Models

An important point is that there is a strong connection between the ranks computed using the surfing and the voting models:

Theorem 5.1 If \mathbf{P} is a column distribution matrix, then the surfer rank fixpoint \mathbf{R}_∞

computed using the (modified) eigenvalue method with exploration probability $\epsilon \in (0, 1)$ and

some start surfers vector \mathbf{S} is equal to the directly assigned vote vector \mathbf{D} computed for

$\mathbf{T} = \epsilon \mathbf{P}$ and $\mathbf{A} = (1 - \epsilon) \mathbf{P}$ with $\mathbf{G} = \mathbf{S}$ and no vote transfer degradation; also in this

case we have that

1.

$$\mathbf{D}_{\%} = (1 - \epsilon) \mathbf{1}$$

2.

$$\mathbf{D} = (1 - \epsilon) \mathbf{C} = (1 - \epsilon) \sum_{i=0}^{\infty} \mathbf{U}_i$$

3.

$$\|\mathbf{C}\|_1 = \frac{|\mathcal{N}|}{(1 - \epsilon)}$$

4.

the surfer rank vector $\mathbf{R} = \mathbf{P} \cdot \mathbf{R}_\infty$ is equal to the vote vector \mathbf{V}_∞

Proof. From $\mathbf{D}_{\%} = \mathbf{A}^T \cdot \mathbf{1}$ we get $\mathbf{D}_{\%} = (1 - \epsilon) \mathbf{P}^T \cdot \mathbf{1}$, but $\mathbf{P}^T \cdot \mathbf{1} = \mathbf{1}$ because

\mathbf{P}^T is a stochastic matrix. This proves item [1](#). Items [2](#) and [3](#) then immediately follow from item [1](#).

To prove the initial statement of the theorem it is enough to show (by uniqueness of the maximal eigenvector) that \mathbf{D} satisfies $\mathbf{D} = \epsilon \mathbf{P} \cdot \mathbf{D} + (1 - \epsilon) \mathbf{S}$. We have that

$$\begin{aligned} \mathbf{D} &= (1 - \epsilon) \sum_{i=0}^{\infty} \mathbf{U}_i \\ &= (1 - \epsilon) \left(\mathbf{I} + \sum_{i=1}^{\infty} \mathbf{T}^i \right) \cdot \mathbf{G} \\ &= (1 - \epsilon) \left(\mathbf{I} + \sum_{i=1}^{\infty} \epsilon^i \mathbf{P}^i \right) \cdot \mathbf{S} \end{aligned}$$

then

$$\begin{aligned}
 & \epsilon \mathbf{P} \cdot \mathbf{D} + (1 - \epsilon) \mathbf{S} \\
 = & \epsilon \mathbf{P} \cdot (1 - \epsilon) (\mathbf{I} + \sum_{i=1}^{\infty} \epsilon^i \mathbf{P}^i) \cdot \mathbf{S} + (1 - \epsilon) \mathbf{S} \\
 = & (1 - \epsilon) (\epsilon \mathbf{P} + \sum_{i=1}^{\infty} \epsilon^{i+1} \mathbf{P}^{i+1}) \cdot \mathbf{S} + (1 - \epsilon) \mathbf{S} \\
 = & (1 - \epsilon) (\mathbf{I} - \mathbf{I} + \sum_{i=1}^{\infty} \epsilon^i \mathbf{P}^i) \cdot \mathbf{S} + (1 - \epsilon) \mathbf{S} \\
 = & (1 - \epsilon) (\mathbf{I} + \sum_{i=1}^{\infty} \epsilon^i \mathbf{P}^i) \cdot \mathbf{S} - (1 - \epsilon) \mathbf{I} \cdot \mathbf{S} + (1 - \epsilon) \mathbf{S} \\
 = & \mathbf{D}
 \end{aligned}$$

Item [4](#) now follows from the following:

$$\mathbf{V}_{\infty} = \mathbf{A} \cdot \sum_{i=0}^{\infty} \mathbf{U}_i = (1 - \epsilon) \mathbf{P} \cdot \sum_{i=0}^{\infty} \mathbf{U}_i = \mathbf{P} \cdot \mathbf{D} = \mathbf{P} \cdot \mathbf{R}_{\infty} .$$

□

This connection between the models [19](#) shows that

- The surfing model is a subcase of the voting model for surfer propagation matrices that are distribution matrices. (This is the natural case, used in the Google search engine [\[1,4\]](#).)
- The exploration probability ϵ is the fraction of the votes (authority ranks) that a node trusts the destinations of its out-links to manage, whereas $1 - \epsilon$ is the fraction of the votes (ranks) a node assigns directly to the destinations of its out-links.
- The rank fixpoint \mathbf{R}_{∞} computed in the surfing model gives the total number of votes

directly assigned by the respective nodes, hence it combines the *overestimation* \mathbf{C} of the measure \mathbf{M} of how trusted a node is (how many votes are trusted to it) and how self-acting the node is (what fraction of the votes it assigns directly rather than delegating the assignment - for the surfing model this fraction is fixed to $1 - \epsilon$). Hence, in the

surfing model \mathbf{R}_{∞} is proportional (with $\frac{1}{1-\epsilon}$) to the overestimation \mathbf{C} of the total number of votes trusted to the respective nodes.

- In the case of the surfing model this trust-assignment split is set for all the nodes uniformly by the model (in the case of the a Web search engine by those who run the rank computation algorithm). In the voting model each node can itself potentially set the trust-assignment split individually for each of its out-links. (In practice this can happen when we have few metadata extensions of the current standards that can be used by Web page authors to specify that trust-assignment split. But it is also possible to extract some hints about the intended trust-assignment split from the text surrounding the links.)

- The surfing model is a bit unnatural in that it corresponds to the voting model where each node trusts some fraction of the votes it has back to itself to redistribute these votes again and again. Hence we can construct a more natural model that uses exactly the same data as the surfing model by letting $\mathbf{T} = \epsilon \mathbf{P}$ and $\mathbf{A} = (1 - \epsilon) \mathbf{P}$ except that we clean \mathbf{T} 's diagonal adding those "self-trusts" to \mathbf{A} 's diagonal.

This connection between the models gives us a very different but well-defined interpretation of the only parameter ϵ and the main results \mathbf{R}_∞ and $\mathbf{P} \cdot \mathbf{R}_\infty$ of the surfing model, allowing one to look at the surfing model from another viewpoint.

Note that it is not clear how to compute all of the statistics of the voting model described in sections 4.3 and 4.4 for the case of the surfing model (an example is the managed vote vector). In general the voting model seems to be better suited for collection of various statistics as it relies on simple direct iterative computation where at each iteration smaller and smaller corrections are added to the result, whereas the surfing model is based on fixpoint iterative computation with nontrivial convergence criteria where at each iteration all the results are recomputed.

6. Extending the Models

6.1 Scores with Confidence

In the voting model the only way a node can directly influence the rank of another node is by changing the fraction of votes it directs to that node. A natural extension to is to let the nodes apply their votes to say that the destination node should have a certain score that reflects its quality on some predefined scale. The node can also specify its confidence in that score estimate (low confidence score estimates will be superseded by high-confidence ones). In this case we can easily compute the overall score estimate of the node as determined by all the votes, and the overall confidence in that score, as well as the total number of votes spent to determine the score. This approach combines the voting model with the way a sports jury or a conference review committee ranks athletes or papers on some absolute scale, except that in our case the relative weight of each jury member is the fraction of votes it has that it wishes to spend to rank the entity in question.

In addition to the vote assignment matrix \mathbf{A} in this case we have the **score assignment matrix** $\mathbf{A}_S = (a_{S_{nn'}})$ such that $a_{S_{nn'}}$ are in say $[0, 1]$ and the **score confidence assignment matrix** $\mathbf{A}_C = (a_{C_{nn'}})$ such that $a_{C_{nn'}} \in [0, 1]$. [20](#)

The score-confidence-vote computation is done by the following formulas:

$$\begin{aligned}
V_0 &= 0 & V_{CV_0} &= 0 & V_{SCV_0} &= 0 \\
U_0 &= G \\
V_{i+1} &= V_i + A \cdot U_i \\
V_{CV_{i+1}} &= V_{CV_i} + (A_C \odot A) \cdot U_i \\
V_{SCV_{i+1}} &= V_{SCV_i} + (A_S \odot A_C \odot A) \cdot U_i \\
U_{i+1} &= T \cdot U_i
\end{aligned}$$

The final **score vector** V_S is then $V_{SCV_\infty} \oslash V_{CV_\infty}$ and the final **confidence vector**

V_C is then $V_{CV_\infty} \oslash V_\infty$, where \oslash is element-wise matrix division. [21](#)

We can see that the score shows the overall community estimate of the "goodness" of the node. The confidence shows the overall explicit confidence in the score as expressed by the voters. The number of votes used to determine the score shows how reliable the score and the confidence are (i.e. how much the node community is interested in getting the right value of the score and the confidence or how much the ranked node is known).

It seems to be impossible to model this extension of the voting model by some extension of the surfing model because in the latter the vote trusts and assignments can not be separated (which is what is needed for the scores with confidence extension) as they are just different fractions of the same matrix.

6.2 Negative Opinions

Another valuable extension is to allow the nodes to express some sort of negative opinions: we can have negative vote assignments (when a node votes against a particular node) and we can have negative trust (when a node gives negated votes to a representative to manage; this corresponds to not supporting *all* the decisions of the representative [22](#)).

6.2.1 Simple Voting Model

For the case of the voting model without the scores and confidence, we can implement negative opinions as follows: Instead of the vote assignment matrix A and the vote trust matrix T , we have the **positive vote assignment matrix** A_+ , the **negative vote assignment matrix** A_- ,

the **positive vote trust matrix** T_+ , and the **inverting vote trust matrix** T_- , such that all

their elements are in $[0, 1]$ and $\sum_{n \in N} (a_{+nn'} + a_{-nn'} + t_{+nn'} + t_{-nn'}) \in [0, 1]$

for any $n' \in N$ (that is, all the votes a node has, it must either distribute between these four

types of voting or waste). The results we are computing then consist of the **positive vote vector**

V_+ and the **negative vote vector** V_- (elements of both of them are always non-negative)

and the computation is done as follows:

$$\begin{aligned}
 V_{+0} &= 0 & V_{-0} &= 0 \\
 U_{+0} &= G & U_{-0} &= 0 \\
 V_{+i+1} &= V_{+i} + A_+ \cdot U_{+i} + A_- \cdot U_{-i} \\
 V_{-i+1} &= V_{-i} + A_- \cdot U_{+i} + A_+ \cdot U_{-i} \\
 U_{+i+1} &= T_+ \cdot U_{+i} + T_- \cdot U_{-i} \\
 U_{-i+1} &= T_- \cdot U_{+i} + T_+ \cdot U_{-i}
 \end{aligned}$$

Then $V_+ - V_-$ gives the absolute number of votes received by the node (it can be positive or negative and can be seen as a vote score) and $V_+ + V_-$ gives the total number of votes that have contributed to the above vote score.

6.2.2 Voting Model with Scores

For the case of the voting model with scores and confidence, we set the score range to say $[-1, 1]$, where 1 corresponds to "completely good node" and -1 , to "absolutely bad node".

Then, instead of A , we have the score assignment matrix A_S divided into the positive one

A_{S+} and the negative one A_{S-} , and the computation is done along the similar lines:

$$\begin{aligned}
 V_0 &= 0 & V_{cv0} &= 0 & V_{scv0} &= 0 \\
 U_{+0} &= G & U_{-0} &= 0 \\
 V_{i+1} &= V_i + A \cdot (U_{+i} + U_{-i}) \\
 V_{cv_{i+1}} &= V_{cv_i} + (A_C \odot A) \cdot (U_{+i} + U_{-i}) \\
 V_{scv_{i+1}} &= V_{scv_i} + ((A_{S+} - A_{S-}) \odot A_C \odot A) \cdot (U_{+i} - U_{-i}) \\
 U_{+i+1} &= T_+ \cdot U_{+i} + T_- \cdot U_{-i} \\
 U_{-i+1} &= T_- \cdot U_{+i} + T_+ \cdot U_{-i}
 \end{aligned}$$

Hence, the negative opinion voting model without scores and confidence corresponds to the one with where the assigned absolute score and confidence are always equal to one.

Note that the negative trust opinions are very strong: the node objects to all decisions of the negatively trusted node. [23](#) These opinions can also be seen as an indication of the fact that the first node believes that the second one got more support in trusted votes than it deserves, hence

the first node directs to it some anti-support. If a node gets more anti-support than support, it ends up hurting the scores of the nodes it was supporting, except for increasing their coverage statistics (i.e. the total number of votes used to determine their scores). Implementing a voting model that permits nodes to also reduce the coverage in such cases seems to be unreasonable: if a node thinks that some scores are wrong and must be corrected, their correction should lead to the increase of their coverage statistics as some votes *were* used to do the correction. The other choice for such node is to just leave the offending scores alone so that they stay wrong but still have small coverage (which reflects their low reliability).

6.2.3 Surfing Model

It turns out that a certain subcase of the voting model with negative opinions but without scores and confidence can be represented in a corresponding extension of the surfing model. We divide the surfer propagation matrix \mathbf{P} into the **positive surfer propagation matrix** \mathbf{P}_+ and the **inverting surfer propagation matrix** \mathbf{P}_- , such that all their elements are non-negative and $\sum_{n \in N} (p_{+nn'} + p_{-nn'}) \in [0, 1]$ for any $n' \in N$. We then compute the **positive rank vector** \mathbf{R}_+ and the **negative rank vector** \mathbf{R}_- as follows:

$$\begin{aligned} \mathbf{R}_{+0} &= \mathbf{S} \\ \mathbf{R}_{-0} &= \mathbf{0} \\ \mathbf{R}_{+i+1} &= \alpha_i (\epsilon (\mathbf{P}_+ \cdot \mathbf{R}_{+i} + \mathbf{P}_- \cdot \mathbf{R}_{-i}) + (1 - \epsilon) \mathbf{S}) \\ \mathbf{R}_{-i+1} &= \alpha_i \epsilon (\mathbf{P}_- \cdot \mathbf{R}_{+i} + \mathbf{P}_+ \cdot \mathbf{R}_{-i}) \end{aligned}$$

where α_i is such as to make $\|\mathbf{R}_{+i+1} + \mathbf{R}_{-i+1}\|_1 = 1$ (α_i will be $\mathbf{1}$ for all i 's if

$\mathbf{P}_+ + \mathbf{P}_-$ is a column distribution matrix).

Again there is a connection between this surfing model and the corresponding voting model: If $\mathbf{P}_+ + \mathbf{P}_-$ is a column distribution matrix, $\mathbf{T}_+ = \epsilon \mathbf{P}_+$, $\mathbf{T}_- = \epsilon \mathbf{P}_-$,

$\mathbf{A}_+ = (1 - \epsilon) \mathbf{P}_+$, $\mathbf{A}_- = (1 - \epsilon) \mathbf{P}_-$, and $\mathbf{G} = \mathbf{S}$, then we have that the vote

score estimate $\mathbf{V}_{+\infty} - \mathbf{V}_{-\infty}$ is equal to $(\mathbf{P}_+ - \mathbf{P}_-) \cdot (\mathbf{R}_{+\infty} - \mathbf{R}_{-\infty})$; the vote

coverage $\mathbf{V}_{+\infty} + \mathbf{V}_{-\infty}$ is equal to $(\mathbf{P}_+ + \mathbf{P}_-) \cdot (\mathbf{R}_{+\infty} + \mathbf{R}_{-\infty})$; and we have

the following equations for the rank fixpoints: $R_{+\infty} = (1 - \epsilon) \sum_{i=0}^{\infty} U_{+i}$,

$$R_{-\infty} = (1 - \epsilon) \sum_{i=0}^{\infty} U_{-i}, \quad R_{+\infty} - R_{-\infty} = (1 - \epsilon) \sum_{i=0}^{\infty} (U_{+i} - U_{-i}).$$

[24](#)

6.3 From Nodes to Documents

In practice the nodes that we have worked with and represent different autonomous authors, are composed of many connected Web pages, hence we have to extend our methods to handle such many-document ``nodes". Below we will concentrate on the needed extensions for the voting model.

The requirements are that

- The number of documents by the same author and the number of times a certain vote trusting, vote assigning, or other opinion is expressed in these documents should not affect the impact they make.
- Other authors should be able to assign votes to (i.e. rank) individual documents.
- It should be easy for the authors to manage the votes they are trusted with, in particular to collect these votes and distribute some of them among their own documents.

The proposed implementation method is as follows:

- When assigning votes the authors indicate the document, hence we will have the assignment of votes to individual documents (they can be easily combined to see how many votes all the documents by a given author have gotten).
- For the purposes of vote (re)trusting we consider all documents by the same author as one virtual document, so that all the trusted votes (including the initial ones) are given to the virtual document as a whole and are assigned or trusted to other authors or assigned to a particular page of the author itself from the virtual document as a whole (hence, neither the number of documents by the same author nor the number of links in them changes the computed ranks).

6.4 Search Engine Persuasion

An important issue of search engine technology is search engine persuasion when the ranks computed by the search engine can be purposefully manipulated by serving certain kind of pages to the search engine. As [\[10\]](#) indicates, link-based ranks are generally much less vulnerable to manipulation than traditional text-based ranks. The previous section addresses the question of rank manipulation by a single author, but for any liberal way to determine the authorship area it is still going to be possible to serve pages corresponding to many fake authors. Thus, we need a way to mitigate the effect of rank manipulation caused by fake authors.

Rank manipulation can be suppressed by down-scaling the initial votes the authors are given according to the number of authors found on hosts with the same (or very similar) IP addresses. [25](#) This method might sometimes down-scale the initial votes of some real non-manipulating authors,

but it is not going to prevent such authors from getting many supporting votes trusted by others. Note that this extension can be implemented along the similar lines for the surfing model by down-scaling certain start surfers vector values.

6.5 Personalized Statistics

The ranks computed using both the voting and the surfing models can be customized by changing the initial number of votes given to the authors (respectively by the number of surfers starting from the documents of certain authors [10]). For example, one can dramatically increase the number of votes (or surfers) given to one's own homepage to get personalized ranks.

The important difference between the two models here is that in the surfing model all the ranks have to be recomputed using the iterative eigenvector computation, whereas in the voting model only the additional votes (i.e. vote differences) have to be propagated along the trust and assignment links to get the differences of the votes assigned to the affected documents. However, the changes will be wider when the effects of vote assignment and especially trust opinions made by the authors whose votes are changed are large.

7. Conclusion

In this paper we have presented a new model, the voting model, for computing various ranks of Web pages to be used by a search engine to better estimate the quality and relevance of the Web pages for various user needs. The new model possesses many advantages over the surfing model that is used in the most successful link-based search engine [Google](#) [4]:

- It subsumes the essential subcase of the surfing model.
- The computation needed for it or its extensions is trivially converging (convergence is crucial to claim meaningfulness of the produced ranks): Only simple finite vote delegation has to be done rather than an iterative eigenvector computation with non-trivial convergence criteria that is needed in the surfing model.
- Due to its simplicity, it allows for more statistics and for easier statistic computation, as well as permits more extensions (e.g. scores with confidence) and allows for faster personalization of the results, which potentially leads to the ability to build a comprehensive search engine service non-trivially personalized for each user.
- It gives all the flexibility to the Web page authors: those who run the rank computation can provide some default values of the parameters influencing the ranks, but the authors can always override them. ²⁶ Whereas the surfing model's important parameter is under sole control of those who run the rank computation.

We have also considered different existing Web page rank computation models under a unifying notion of rank flows, as well as analyzed the surfing model of Google. In particular using the proposed voting model we gave another meaning of the parameters and the results of the surfing model and developed its extension to handle links signifying disapproval. We have also covered the issue of search engine persuasion for both models.

Although the true power of the proposed voting model requires various metadata that are

currently not present on Web pages, we believe that it is important to study useful extensions of Web metadata standards and new techniques to exploit them, so as to radically improve the quality of information retrieval services on the Web. In particular we are working on a proposal to extend metadata of HTML/XML and the corresponding search engine infrastructure to support the ability of Web page authors to specify their ``opinions'' about documents referenced in their pages, as well as on methods to extract such data from existing Web pages. We are also working on extending the proposed voting model to support an Internet document taxonomy that itself can be created by Web page authors in a decentralized fashion.

The key advantage of the voting model is that it supports both the surfer model that has been proven today and a more general metadata standard that could potentially further enhance the effectiveness of Web page retrieval in a single framework, and thus provides a smooth migration path for search engine implementations to move the Web towards a much more organized and valuable information service.

Acknowledgements

The research presented in this paper was partially supported by NSF grant IRI-9711635. The author would also like to thank Tzi-cker Chiueh for his support, insightful discussions, and comments on the early versions of this paper.

References

[1]

Sergey Brin and Lawrence Page.

The anatomy of a large-scale hypertextual Web search engine.

In *Proceedings of 7th International World Wide Web Conference*, 14-18 April 1998.

Available at

<http://www7.scu.edu.au/programme/fullpapers/1921/com1921.htm>.

[2]

Chakrabarti, Dom, Kumar, Raghavan, Rajagopalan, Tomkins, Gibson, and Kleinberg.

Mining the link structure of the World Wide Web.

IEEE Computer, 32(8), August 1999.

Available at <http://www.almaden.ibm.com/cs/k53/ieeecomps.ps>.

[3]

Soumen Chakrabarti, Byron Dom, Prabhakar Raghavan, Sridhar Rajagopalan, David Gibson, and Jon Kleinberg.

Automatic resource compilation by analyzing hyperlink structure and associated text.

In *Proceedings of 7th International World Wide Web Conference*, 14-18 April 1998.

Available at

<http://www7.scu.edu.au/programme/fullpapers/1898/com1898.html>.

[4]

Google, <http://www.google.com/>.

[5]

Jon M. Kleinberg.

Authoritative sources in a hyperlinked environment.

In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 668-677, San Francisco, California, 25-27 January 1998.

Available at <http://www.cs.cornell.edu/home/kleinber/auth.ps>.

[6]

Steve Lawrence and C. Lee Giles.

Searching the Web: General and scientific information access.

IEEE Communications, 37(1):116-122, 1999.

Available at

<http://www.neci.nec.com/~lawrence/papers/search-ieee99/>.

[7]

Massimo Marchiori.

The quest for correct information on the Web: Hyper search engines.

In *Proceedings of the 6th International World Wide Web Conference*, Santa Clara, California, April 1997.

Available at <http://www.w3.org/People/Massimo/papers/WWW6/>.

[8]

Massimo Marchiori.

Enhancing navigation in the World Wide Web.

In *SAC '98 - 1998 ACM Symposium on Applied Computing*, Atlanta, Georgia, U.S.A., February 1998.

Available at <http://www.w3.org/People/Massimo/papers/wwwSAC.ps.gz>.

[9]

Hehryk Minc.

Nonnegative matrices.

JWAS, 1988.

[10]

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd.

The pagerank citation ranking: Bringing order to the Web.

Technical report, Stanford University, 1998.

Available at <http://www-db.stanford.edu/~backrub/pageranksub.ps>,

slides available at

<http://www-pcd.stanford.edu/~page/papers/pagerank/>.

Vitae



[Maxim Lifantsev](#) is currently a Ph.D. student in the Department of Computer Science at the State University of New York at Stony Brook, U.S.A. His research interests include innovative scalable Web services, distributed collaborative Web categorization and ranking, mining and enhancing the link structure of the Web, integration of search engines and personal browsing assistants. He received his diploma (M.S.) in applied mathematics from Moscow State Engineering-Physics Institute, Russia in 1996.

Footnotes

...[\[1\]](#).¹

Actually, this flow can be subdivided into two types: one is when page **A** assigns some of its authority to **B** stating that **B** is good, and the other one, when **A** gives some of its authority to **B** to use as **B** wishes. This division is the foundation of the voting model proposed in section [4](#).

... **A** [2](#)

But if there are too many links in **A**, then it can be difficult for the user to locate (hence "include") page **B** [\[7\]](#). A personal assistant augmenting Web pages can help the user here.

... pages.³

See [\[7\]](#) for a proposal on how to fade this rank flow when we traverse more than one link.

... influence.⁴

The idea of this flow seems to be new, though later we will see that the ranks defined by the PageRank method of Google [\[10\]](#) actually correspond more to authority determination ranks rather than to authority ranks.

... manipulations.⁵

This has not been a real problem as there are no widely used implementations of Clever, hence there is no incentive for such rank manipulations.

... analysis.⁶

The idea of author incitement using search engine bonuses was proposed in [\[8\]](#).

... surfers⁷

$|N|$ is the number of nodes in **N**.

... incomplete⁸

In the sense that the probabilities sum up to less than one.

... literature).⁹

It is natural to have \mathbf{P} as a column distribution matrix, also an incomplete column distribution matrix can be naturally completed by adding the missing fractions to the corresponding diagonal elements.

... configuration:¹⁰

We will use \cdot for regular matrix multiplication.

... $\dot{\mathbf{v}}$.¹¹

$\|\mathbf{B}\|_1$ is the **1-norm** for vectors and matrices: $\|\mathbf{B}\|_1 = \sum_{i,j} |b_{ij}|$ for

$$\mathbf{B} = (b_{ij}).$$

... $\mathbf{1}$.¹²

Clearly this computation yields the same results as the unmodified method (1) when \mathbf{P} is a column distribution matrix.

... manage.¹³

It is also possible to have a model where e.g. a node can specify how many times the votes it trusts can be "retrusted". We do not consider such models here as they do not seem to add much value over the basic model.

... given.¹⁴

It is natural to have \mathbf{A} and \mathbf{T} as distribution matrices; also incomplete vote distribution matrices can be naturally completed by adding the missing fractions to the corresponding diagonal elements of \mathbf{A} .

... node.¹⁵

\odot is the element-wise matrix multiplication, also known as Hadamard or Schur multiplication.

... cycles¹⁶

This is ensured by the zeroing out of the diagonal at each step.

... distribution.¹⁷

In the voting model we can say that a node is a good hub if it has high trusted vote vector value, but low vote vector value: the node is highly trusted but passes all the votes to the authorities.

... with.¹⁸

We can also compute a different set of statistics that might be slightly better suited for this

purpose if we progressively down-scale the differences between \mathbf{M}_i and \mathbf{M}_{i+1} .

... models¹⁹

Note that this theorem does not hold when \mathbf{P} is an incomplete column distribution matrix: The ranks generated by both methods with the stated relationships in the input data are not even proportional to each other. This is because in the voting model wasted votes just disappear, whereas in the surfing model undirected surfers are recreated in a way dependent on the current ranks.

... $a_{cn} \in [0, 1]$.²⁰

In fact, we can even let the nodes specify many score assignments with different vote percents and confidence for the same destination node.

... division.²¹

If we assume that $0/0 = 0$, then \mathbf{V}_s and \mathbf{V}_c are always well-defined and have finite elements.

... representative²²

The same way as the regular trust corresponds to supporting *all* the decisions of the representative.

... node.²³

It might be good to also implement some partial negative trusts: e.g. when the negatively trusted votes affect only all the direct assignments (or the assignments and only all the positive trusts) of the node receiving them.

... $\mathbf{R}_{+\infty} - \mathbf{R}_{-\infty} = (1 - \epsilon) \sum_{i=0}^{\infty} (\mathbf{U}_{+i} - \mathbf{U}_{-i})$.²⁴

The formal proof of these results should be similar to the proof of theorem [5.1](#) and is left as an exercise.

... addresses.²⁵

The assumption is that a manipulator can not control many (physical) hosts with different IP's.

... them.²⁶

When they have a metadata standard for that, which is supported by the search engine computing the ranks. Arguing necessity of such a standard is beyond the scope of this paper.
