Finding Authorities and Hubs From Link Structures on the World Wide Web *

Allan Borodin† Gareth O. Roberts‡ Jeffrey S. Rosenthal§ Panayiotis Tsaparas¶

November 9, 2001

Abstract

Recently, there have been a number of algorithms proposed for analyzing hypertext link structure so as to determine the best "authorities" for a given topic or query. While such analysis is usually combined with content analysis, there is a sense in which some algorithms are deemed to be "more balanced" and others "more focused". We undertake a comparative study of hypertext link analysis algorithms. Guided by some experimental queries, we propose some formal criteria for evaluating and comparing link analysis algorithms.

Keywords: link analysis, web searching, hubs, authorities, SALSA, Kleinberg's algorithm, threshold, Bayesian.

1 Introduction

In recent years, a number of papers [7, 14, 5, 19, 15, 8] have considered the use of hypertext links to determine the value of different web pages. In particular, these papers consider the extent to which hypertext links between World Wide Web documents can be used to determine the relative authority values of these documents for various search queries.

We consider some of the previously published algorithms as well as introducing some new alternatives. One of our new algorithms is based on a Bayesian statistical approach as opposed to the more common algebraic/graph theoretic approach. While link analysis by itself cannot be expected to always provide reliable rankings, it is interesting to study various link analysis strategies in an attempt to understand inherent limitations, basic properties and "similarities" between the various methods. To this end, we offer definitions for several intuitive concepts relating to (link analysis) ranking algorithms and begin a study of these concepts.

We also provide some new (comparative) experimental studies of the performance of the various ranking algorithms. It can be seen that no method is completely safe from "topic drift", but some methods do seem to be more resistant than others. We shall see that certain methods have surprisingly similar rankings as observed in our experimental studies, however they cannot be said to be similar with regard to our formalization.

*A preliminary version of this paper has appeared in the Proceedings of the 10th International World Wide Web Conference.

†Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 3G4 and Gammasite, Hertzilya, Israel. E-mail: bor@cs.toronto.edu. Web: http://www.cs.utoronto.ca/DCS/People/Faculty/bor.html.

‡Department of Mathematics and Statistics, Lancaster University, Lancaster, U.K. LA1 4YF. E-mail: g.o.roberts@lancaster.ac.uk. Web: http://www.maths.lancs.ac.uk/dept/people/robertsg.html.

§Department of Statistics, University of Toronto, Toronto, Ontario, Canada M5S 3G3. Supported in part by NSERC of Canada. E-mail: jeff@math.toronto.edu. Web: http://markov.utstat.toronto.edu/jeff/.

¶Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 3G4. E-mail: tsap@cs.toronto.edu. Web: http://www.cs.toronto.edu/~tsap/.

2 Previous Algorithms

2.1 The PageRank Algorithm

One of the earliest and most commercially successful of the efforts to use hypertext link structures in web searching is the PageRank algorithm used by Brin and Page [7] in the Google search engine [12]. The PageRank algorithm is query independent, that is, it operates on the whole Web, and assigns a PageRank value to every page. The PageRank of a given web page i, $PR(i)$, can be defined as the limiting fraction of time spent on page i by a random walk which proceeds at each step as follows: With probability ϵ it jumps to a sample from a distribution $D(\cdot)$ (e.g. the uniform distribution) , and with probability $1-\epsilon$ it jumps uniformly at random to one of the pages linked from the current page. This idea is also used by Rafiei and Mendelzon [19] for computing the "reputation" of a page. Intuitively, the value of $PR(i)$ is a measure of the importance or authority of the web page i. This ranking is used as one component of the Google search engine, to help determine how to order the pages returned by a web search query.

2.2 Kleinberg's Algorithm

Independent of Brin and Page, Kleinberg [14] proposed a more refined notion for the importance of web pages. He suggested that web page importance should depend on the search query being performed. Furthermore, each page should have a separate "authority" rating (based on the links going *to* the page) and "hub" rating (based on the links going *from* the page). Kleinberg proposed first using a text-based web search engine (such as AltaVista [2]) to get a *Root Set* consisting of a short list of web pages relevant to a given query. Second, the Root Set is augmented by pages which link to pages in the Root Set, and also pages which are linked to pages in the Root Set, to obtain a larger *Base Set* of web pages. If N is the number of pages in the final Base Set, then the data for Kleinberg's algorithm consists of an $N \times N$ adjacency matrix A, where $A_{ij} = 1$ if there are one or more hypertext links from page i to page j, otherwise $A_{ij} = 0$.

Kleinberg's algorithm assigns to each page i an authority weight a_i and a hub weight h_i. Let $\boldsymbol{a} = (a_1, a_2, \ldots, a_N)$ denote the vector of all authority weights, and $\boldsymbol{h} = (h_1, h_2, \ldots, h_N)$ the vector of all hub weights. Initially both authority and hub vectors are set to $\boldsymbol{u}= (1, 1, \ldots, 1)$. At each iteration the operations $\mathcal{I}$ ("in") and $\mathcal{O}$ ("out") are performed. The operation $\mathcal{I}$ sets the authority vector to $\boldsymbol{a} = A^T\ \boldsymbol{h}$. The operation $\mathcal{O}$ sets the hub vector to $\boldsymbol{h} = A\ \boldsymbol{a}$. A normalization step is then applied, so that the vectors $\boldsymbol{a}$ and $\boldsymbol{h}$ become unit vectors in some norm. Kleinberg proves that after a sufficient number of iterations the vectors $\boldsymbol{a}$ and $\boldsymbol{h}$ converge to the principal eigenvectors of the matrices A^TA and AA^T, respectively. The above normalization step may be performed in various ways. Indeed, *ratios* such as a_i/a_j will converge to the same value no matter how (or if) normalization is performed.

Kleinberg's Algorithm (and some of the other algorithms we are considering) converge naturally to their principal eigenvector, i.e. to the eigenvector that corresponds to the largest eigenvalue of a matrix associated with the algorithm. Kleinberg [14] makes an interesting (though non-precise) claim that the secondary *non-principal* eigenvectors (or their positive and negative components) are sometimes representative of "sub-communities" of web pages. It is easy to construct simple examples which show that secondary eigenvectors sometimes are, but sometimes are not, indicative of sub-communities. We present a few indicative such examples in section 5.

2.3 The SALSA Algorithm

An alternative algorithm, SALSA, was proposed by Lempel and Moran [15]. Like Kleinberg's algorithm, SALSA starts with a similarly constructed Base Set. It then performs a random walk by alternately (a) going uniformly to one of the pages which links to the current page, and (b) going uniformly to one of the pages linked to by the current page. The authority weights are defined to be the stationary distribution of the two-step chain doing first step (a) and then (b), while the hub weights are defined to be the stationary distribution of the two-step chain doing first step (b) and then (a).

Formally, let $B(i) = \{k : k \rightarrow i\}$ denote the set of all nodes that point to i, that is, the nodes we can reach from i by following a link backwards, and let $F(i) = \{k : i \rightarrow k\}$ denote the set of all nodes that we can reach from i by following a forward link. The Markov Chain for the authorities has transition probabilities

$$P_a(i,j) = \sum_{k\,:\,k \in B(i) \cap B(j)} \frac{1}{|B(i)|} \frac{1}{|F(k)|}.$$

This Markov Chain corresponds to a random walk on the *authority* graph G_a, defined by the adjacency matrix $A^T A$, where we move from authority i to authority j with probability $P_a(i,j)$.

Assume for a moment that the Markov Chain is *irreducible*, that is, the underlying authority graph consists of a single component, where we can move between any two authorities, by following a backward and a forward link. The authors prove that the stationary distribution $\boldsymbol{a} = (a_1, a_2, ..., a_N)$ of the Markov Chain satisfies $a_i = |B(i)| \; / \; |B|$, where $B = \bigcup_i B(i)$ is the set of all (backward) links.

A similar Markov Chain is defined for the hubs, that has transition probabilities

$$P_h(i,j) = \sum_{k\,:\,k \in F(i) \cap F(j)} \frac{1}{|F(i)|} \frac{1}{|B(k)|},$$

and the stationary distribution $\boldsymbol{h} = (h_1, h_2, ..., h_N)$ satisfies $h_i = |F(i)| \; / \; |F|$, where $F = \bigcup_i F(i)$ is the set of all (forward) links.

The SALSA algorithm does not really have the same "mutually reinforcing structure" that Kleinberg's algorithm does. Indeed, since $a_i = |B(i)|/|B|$, the relative authority of site i *within a connected component* is determined from local links, not from the structure of the component. (See also the discussion of *locality* in Section 8.) We also note that in the special case of a single component, SALSA can be viewed as a one-step truncated version of Kleinberg's algorithm. That is, in the first iteration of Kleinberg's algorithm, if we perform the $\mathcal{I}$ operation first, the authority weights are set to $\boldsymbol{a} = A^T \, \boldsymbol{u}$, where $\boldsymbol{u}$ is the vector of all ones. If we normalize in the L_1 norm, then $a_i = \frac{|B(i)|}{|B|}$, which is the stationary distribution of the SALSA algorithm. A similar observation can be made for the hub weights.

If the underlying authority graph G_a consists of more than one component, then the SALSA algorithm selects a starting point uniformly at random, and performs a random walk within the connected component that contains that node. Formally, let j be a component that contains node i, let A_j denote the number of authorities in the component j, and B_j the set of (backward) links in component j. Also, let A denote the total number of authorities in the graph (a node is an authority only if it has non-zero in-degree). Then the weight of authority i is

$$a_i = \frac{A_j}{A} \frac{|B(i)|}{|B_j|} \; .$$

Motivated by the simplifying assumption of a single component, in the conference version of this paper [6], we considered a simplified version of the SALSA algorithm where the authority weight of node i is the ratio $|B(i)|/|B|$. This corresponds to the case that the starting point for the random walk is chosen with probability proportional to the "popularity" of the node, that is, the number of links that point to this node. We will refer to this variation of the SALSA algorithm as pSALSA (popularity SALSA). We will also consider the original SALSA algorithm as defined in [15]. When the distinction between pSALSA and SALSA is not important we will use the name SALSA to collectively refer to both algorithms.

An interesting generalization of the SALSA algorithm is considered by Rafiei and Mendelzon [19]. They propose an algorithm for computing reputations that is a hybrid of the SALSA algorithm, and the PageRank algorithm. At each step, with probability ϵ, the Rafiei and Mendelzon algorithm jumps to a page of the collection chosen uniformly at random, and with probability $1 - \epsilon$ it performs a SALSA step. This algorithm is essentially the same as the *Randomized HITS* algorithm considered later by Ng et al. [18].

2.4 The PHITS Algorithm

Cohn and Chang [8] propose a statistical hubs and authorities algorithm, which they call the PHITS Algorithm. They propose a probabilistic model in which a citation c of a document d is caused by a latent "factor" or "topic", z. It is postulated that there are conditional distributions $P(c|z)$ of a citation c given a factor z, and also conditional distributions $P(z|d)$ of a factor z given a document d. In terms of these conditional distributions, they produce a *likelihood function*.

Cohn and Chang then propose using the EM Algorithm of Dempster et al. [9] to assign the unknown conditional probabilities so as to maximize this likelihood function L, and thus best "explain" the proposed data. Their algorithm requires specifying in advance the number of factors z to be considered. Furthermore, it is possible that their EM Algorithm could get "stuck" in a local maximum, without converging to the true global maximum.

3 Random Walks and the Kleinberg Algorithm

The fact that the output of the first (half) step of the Kleinberg algorithm can be seen as the stationary distribution of a certain random walk on the underlying graph, poses the natural question of whether other intermediary results of Kleinberg's algorithm (and as $n \to \infty$, the output of the algorithm itself) can also be seen as the stationary distribution of a *naturally* defined random walk [1]. We will show that this is indeed the case.

We first introduce the following notation. We say that we follow a B path if we follow a link backwards, and we say we follow an F path if we follow a link forward. We can combine these to obtain longer paths. For example, a $(BF)^n$ path is a path that alternates between backward and forward links n times. Now, let $(BF)^n(i,j)$ denote the set of $(BF)^n$ paths that go from i to j, $(BF)^n(i)$ the set of $(BF)^n$ paths that leave node i, and $(BF)^n$ the set of all possible $(BF)^n$ paths. We can define similar sets for the $(FB)^n$ paths.

Now, we define the undirected weighted graph $G_{(BF)^n}$ as follows. The vertex set of the graph is the set of nodes in the base set. We place an edge between two nodes i and j if there is a $(BF)^n$ path between these nodes. The weight of the edge is $|(BF)^n(i,j)|$, the number of $(BF)^n$ paths between i and j. We perform a random walk on graph $G_{(BF)^n}$. When at node i, we move to node j with probability proportional to the number of paths between i and j. The corresponding Markov Chain $M_{(BF)^n}$ has transition probabilities

$$P_a(i,j) = \frac{|(BF)^n(i,j)|}{|(BF)^n(i)|}.$$

Similarly, we can define the graph $G_{(FB)^n}$, and the corresponding Markov Chain $M_{(FB)^n}$, for the hubs case.

Theorem 1 *For each $n \geq 1$, the stationary distribution of $M_{(BF)^n}$ is equal to the authority vector after the n^{th} iteration of the Kleinberg algorithm, and the stationary distribution of $M_{(FB)^n}$ is equal to the hub vector after the n^{th} iteration of the Kleinberg algorithm.*

Proof: By definition of the $(A^TA)^n$, and $(AA^T)^n$ matrices, we have that $|(BF)^n(i,j)| = (A^TA)^n(i,j)$, and $|(FB)^n(i,j)| = (AA^T)^n(i,j)$. Also, $|(BF)^n(i)| = \sum_j (A^TA)^n(i,j)$, and $|(FB)^n(i)| = \sum_j (AA^T)^n(i,j)$. After the n^{th} operation of the Kleinberg algorithm the authority vector $\boldsymbol{a}$, and hub vector $\boldsymbol{h}$ are the unit vectors in the direction of $(A^TA)^n\boldsymbol{u}$ and $(AA^T)^n\boldsymbol{u}$, respectively. (This actually assumes that in order to compute the authority weights we switch the order of the operations $\mathcal{I}$ and $\mathcal{O}$, but asymptotically this does not make any difference). If we take the unit vectors under the L_1 norm, then we have

$$a_i = \frac{|(BF)^n(i)|}{|(BF)^n|}, \qquad \text{and} \qquad h_i = \frac{|(FB)^n(i)|}{|(FB)^n|}. \tag{1}$$

From a standard theorem on random walks on weighted graphs (see, e.g., p. 132 of [16] for the corresponding result on unweighted graphs), the stationary distribution of the Markov Chain $M_{(BF)^n}$ is the same as the vector $\boldsymbol{a}$ in equation (1), while the stationary distribution of the Markov Chain $M_{(FB)^n}$ is the same as the vector $\boldsymbol{h}$ in the same equation. □

[1] It is easy to show that for any probability vector $\mathbf{p}$, there exists a Markov Chain M, such that $\mathbf{p}$ is the stationary distribution of M. Here, naturally defined Markov Chain means a Markov Chain that is related to the underlying graph of the algorithm.

4 Some modifications to the Kleinberg and SALSA Algorithms

While Kleinberg's algorithm has some very desirable properties, it also has its limitations. One potential problem is the possibility of severe "topic drift". Roughly, Kleinberg's algorithm converges to the most "tightly-knit" community within the Base Set. It is possible that this tightly-knit community will have little or nothing to do with the proposed query topic.

A striking example of this phenomenon is provided by Cohn and Chang ([8], p. 6). They use Kleinberg's Algorithm with the search term "jaguar" (an example query suggested by Kleinberg [14]), and converge to a collection of sites about the city of Cincinnati! They determine that the cause of this is a large number of on-line newspaper articles in the Cincinnati Enquirer which discuss the Jacksonville Jaguars football team, and all link to the same standard Cincinnati Enquirer service pages. Interestingly, in a preliminary experiment with the query term "abortion" (another example query suggested by Kleinberg [14]), we also found the Kleinberg Algorithm converging to a collection of web pages about the city of Cincinnati!

Now, in both these cases, we believe it is possible to eliminate such errant behavior through more careful selection of the Base Set, and more careful elimination of intra-domain hypertext links. Nevertheless, we do feel that these examples point to a certain "instability" of Kleinberg's Algorithm.

4.1 The Hub-Averaging Kleinberg Algorithm

We propose here a small modification of Kleinberg's algorithm to help remedy the above-mentioned instability. For motivation, consider the following example. Suppose there are $K+1$ authority pages, and $M+1$ hub pages, with M and K large. The first hub points to all but the first authority (i.e. to the final K authorities). The next $M-1$ hubs link only to the first authority. The last hub points to the first two authorities, and serves the purpose of connecting the graph. In such a set-up, we would expect the first authority to be considered much more authoritative than all the others. However, if K and M are chosen appropriately, the Kleinberg algorithm allocates almost all authority weight to the last K authorities, and almost no weight to the first authority. This is due to the fact that almost all of the hub weight is allocated to the first hub. It seems though that the first hub should be *worse* than the others, since it links only to "bad" authorities (in the sense that no other hub points to them).

Inspired by such considerations, we propose an algorithm which is a "hybrid" of the Kleinberg and SALSA algorithms. Namely, it does the authority rating updates $\mathcal{I}$ just like Kleinberg (i.e., giving each authority a rating equal to the sum of the hub ratings of all the pages that link to it), but does the hub rating updates $\mathcal{O}$ by instead giving each hub a rating equal to the *average* of the authority ratings of all the pages that it links to. This asymmetric view of hubs and authorities is corroborated by the observation that in contrast to the in degree which gives an indication of the quality of a node as an authority, the out degree is less informative when assessing the quality of a node as a hub. In the Kleinberg algorithm a hub can increase its weight simply by pointing to more nodes in the graph. In this modified "Hub-Averaging" algorithm, a hub is better if it links to *only* good authorities, rather than linking to both good *and* bad authorities.

4.2 The Threshold Kleinberg Algorithms

We propose two different "threshold" modifications to Kleinberg's Algorithm. The first modification, Hub-Threshold, is applied to the in-step $\mathcal{I}$. When computing the authority weight of page i, the algorithm does not take into account all hubs that point to page i. It only counts those hubs whose hub weight is at least the average hub weight [2] over all the hubs that point to page i, computed using the current hub weights for the nodes. This corresponds to saying that a site should not be considered a good authority simply because a lot of very poor hubs point to it.

The second modification, Authority-Threshold, is applied to the out-step $\mathcal{O}$. When computing the hub weight of page i, the algorithm does not take into account all authorities pointed to by page i. It only counts those authorities which are among the top K authorities, judging by current authority values. The value of K is passed as a parameter to the algorithm. This corresponds to saying that a site should not be considered a good hub simply because it points

[2] Other thresholds are also possible. For example the median hub weight, or $(1-\delta)w_{max}$, where w_{max} is the maximum hub weight over all hubs that point to the authority, and $0 < \delta < 1$.

to a number of "acceptable" authorities; rather, to be considered a good hub the site must point to some of the *best* authorities. This is inspired partially by the fact that, in most web searches, a user only visits the top few authorities.

We note that if $K = 1$, then we transform the $\mathcal{O}$ operation to the max operator. The case $K = 1$ has some interesting properties. It is not hard to see that the node with the highest in-degree will always be ranked first. The rest of the nodes are ranked depending on the amount of connectivity with, and the distance to the top node. Therefore, in this case the most popular node acts as a a *seed* to the algorithm: this node is ranked first, and the rest of the nodes are ranked according to their relatedness to this node.

We also consider a Full-Threshold algorithm, which makes *both* the Hub-Threshold and Authority-Threshold modifications to Kleinberg's Algorithm.

4.3 The Breadth-First-Search Algorithm: A Normalized n-step Variant

When the pSALSA algorithm computes the authority weight of a page, it takes into account only the popularity of this page within its immediate neighborhood, disregarding the rest of the graph. On the other hand, the Kleinberg algorithm considers the whole graph, taking into account more the structure of the graph around the node, than just the popularity of that node in the graph. Specifically, after n steps, the authority weight of authority i is $|(BF)^n(i)|/|(BF)^n|$, where $|(BF)^n(i)|$ is the number of $(BF)^n$ paths that leave node i. Another way to think of this is that the contribution of a node $j \neq i$ to the weight of i is equal to the number of $(BF)^n$ paths that go from i to j. Therefore, if a small bipartite component intercepts the path between node j and i, the contribution of node j will increase exponentially fast. This may not always be desirable, especially if the bipartite component is not representative of the query.

We propose the Breadth-First-Search (BFS) algorithm, as a generalization of the pSALSA algorithm, and a restriction of the Kleinberg algorithm. The BFS algorithm extends the idea of popularity that appears in pSALSA from a one link neighborhood to an n-link neighborhood. The construction of the n-link neighborhood is inspired by the Kleinberg algorithm. However, instead of considering the number of $(BF)^n$ *paths* that leave i, it considers the number of $(BF)^n$ *neighbors* of node i. Abusing the notation, let $(BF)^n(i)$ denote the set of nodes that can be reached from i by following a $(BF)^n$ path. The contribution of node j to the weight of node i depends on the distance of the node j from i. We adopt an exponentially decreasing weighting scheme. Therefore, the weight of node i is determined as follows:

$$a_i = |B(i)| + \frac{1}{2}|BF(i)| + \frac{1}{2^2}|BFB(i)| + \ldots + \frac{1}{2^{2n-1}}|(BF)^n(i)|.$$

The algorithm starts from node i, and visits its neighbors in BFS order, alternating between Backward and Forward steps. Every time we move one link further from the starting node i, we update the weight factors accordingly. The algorithm stops when n links have been traversed, or the nodes that can be reached from node i are exhausted.

5 Secondary Eigenvectors in the Kleinberg Algorithm

The Kleinberg Algorithm (and many of the other algorithms we are considering) converge naturally to the principal eigenvector of the associated matrix, i.e. to the eigenvector that corresponds to the largest eigenvalue. Kleinberg [14] makes an interesting (though non-precise) claim regarding the *secondary* (i.e., non-principal) eigenvectors (or their positive and negative components) being related to secondary (or opposing) "communities" of web pages. The use of secondary eigenvectors for discovering communities, or for improving the quality of the ranking has been investigated further in [13, 1, 18].

We now present a few simple examples which we feel illustrate the opinion that such secondary eigenvectors sometimes are, but sometimes are not, indicative of secondary communities. In short, there is no simple result either way, regarding these secondary eigenvectors. For the following, we write the examples as a sequence of two-digit numbers, where each number represents a link, with the first digit being the hub number and the second digit being the authority number. For example, 23, 24, 34 indicates that there are links between the second hub and third authority; second hub and fourth authority; and third hub and fourth authority. (All the examples have fewer than 10 hubs and fewer than 10 authorities, so this notation will suffice for now.)

Example E1: 11, 21, 31, 41, 52, 62, 53, 63

The corresponding matrix of transitions of authority weights is given by

$$AA^T = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 2 & 2 \\ 0 & 2 & 2 \end{pmatrix}.$$

The eigenvalues of the matrix are 4, 4, and 0. Here the equality of two eigenvalues means that we have a wide choice of how to choose representative eigenvectors. One possible choice for the eigenvectors is (0,1,1), (1,0,0), (0,1,-1). In this case, there is some correspondence between eigenvectors and communities. However, if the eigenvectors are chosen to be (1,1,1), (2,1,1), (0,1,-1), there is no correspondence whatsoever.

Example E2: 11, 22, 32, 42, 52, 43, 53, 63, 73

The corresponding matrix of transitions of authority weights is given by

$$AA^T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 4 & 2 \\ 0 & 2 & 4 \end{pmatrix}.$$

The eigenvalues of the matrix are 6, 2, and 1, with corresponding eigenvectors (0,1,1), (0,1,-1), (1,0,0). Here the first and third eigenvectors correspond nicely to communities, but the second eigenvector does not in any way.

These simple examples suggest that:

- Eigenvector components may or may not correspond to communities.
- The eigenvectors which do correspond to communities may not be the first ones (cf. E2).
- Equality of eigenvalues complicates things still further (cf. E1).

Of course, these examples are *reducible*, whereas a more realistic example might be almost-but-not-quite reducible. However, it seems that making the graph irreducible can only make things worse. In particular, the eigenvector weights assigned to two almost-but-not-quite-disjoint pieces can vary widely based on the exact details of the few links joining them. So, we expect the values of the secondary eigenvectors to be even less indicative when partitioning the pages into communities. Thus, it seems to us that there is no clear, simple, rigorous result available about secondary eigenvectors corresponding to secondary communities.

6 A Bayesian Algorithm

A different type of algorithm is given by a fully Bayesian statistical approach to authorities and hubs. Suppose there are M hubs and N authorities (which could be the same set). We suppose that each hub i has an (unknown) real parameter e_i, corresponding to its "general tendency to have hypertext links", and also an (unknown) non-negative parameter h_i, corresponding to its "tendency to have *intelligent* hypertext links to *authoritative* sites". We further suppose that each authority j has an (unknown) non-negative parameter a_j, corresponding to its level of authority.

Our statistical model is as follows. The *a priori* probability of a link from hub i to authority j is given by

$$\mathbf{P}(i \to j) = \frac{\exp(a_j h_i + e_i)}{1 + \exp(a_j h_i + e_i)}, \tag{2}$$

with the probability of no link from i to j given by

$$\mathbf{P}(i \not\to j) = \frac{1}{1 + \exp(a_j h_i + e_i)}. \tag{3}$$

This reflects the idea that a link is more likely if e_i is large (in which case hub i has large tendency to link to *any* site), or if *both* h_i and a_j are large (in which case i is an intelligent hub, and j is a high-quality authority).

To complete the specification of the statistical model from a Bayesian point of view (see, e.g., Bernardo and Smith [4]), we must assign *prior* distributions to the $2M+N$ unknown parameters e_i, h_i, and a_j. These priors should be general and uninformative, and should *not* depend on the observed data. For large graphs, the choice of priors should have only a small impact on the results. We let $\mu = -5.0$ and $\sigma = 0.1$ be fixed parameters, and let each e_i have prior distribution $N(\mu, \sigma^2)$, a normal distribution with mean μ and variance σ^2. We further let each h_i and a_j have prior distribution Exp(1) (since they have to be non-negative), meaning that for $x \geq 0$, $\mathbf{P}(h_i \geq x) = \mathbf{P}(a_j \geq x) = \exp(-x)$.

The (standard) Bayesian inference method then proceeds from this fully-specified statistical model, by *conditioning* on the observed data, which in this case is the matrix A of actual observed hypertext links in the Base Set. Specifically, when we condition on the data A we obtain a *posterior density* $\pi : \mathbf{R}^{2M+N} \to [0, \infty)$ for the parameters $(e_1, \ldots, e_M, h_1, \ldots, h_M, a_1, \ldots, a_N)$. This density is defined so that

$$\begin{aligned} &\mathbf{P}\Big((e_1, \ldots, e_M, h_1, \ldots, h_M, a_1, \ldots, a_N) \in S \;\Big|\; \{A_{ij}\}\Big) \\ &\quad = \int_S \pi(e_1, \ldots, e_M, h_1, \ldots, h_M, a_1, \ldots, a_N) de_1 \ldots de_M dh_1 \ldots dh_M da_1 \ldots da_N \end{aligned} \tag{4}$$

for any (measurable) subset $S \subseteq \mathbf{R}^{2M+N}$, and also

$$\begin{aligned} &\mathbf{E}\Big(g(e_1, \ldots, e_M, h_1, \ldots, h_M, a_1, \ldots, a_N) \;\Big|\; \{A_{ij}\}\Big) \\ &\quad = \int_{\mathbf{R}^{2M+N}} g(e_1, \ldots, e_M, h_1, \ldots, h_M, a_1, \ldots, a_N) \pi(e_1, \ldots, e_M, h_1, \ldots, h_M, a_1, \ldots, a_N) de_1 \ldots de_M dh_1 \ldots dh_M da_1 \ldots da_N \end{aligned}$$

for any (measurable) function $g : \mathbf{R}^{2M+N} \to \mathbf{R}$. An easy computation gives the following.

Lemma 1 *For our model, the posterior density is given, up to a multiplicative constant, by*

$$\begin{aligned} &\pi(e_1, \ldots, e_M, h_1, \ldots, h_M, a_1, \ldots, a_N) \\ &\propto \prod_{i=1}^{M} \exp(-h_i) \exp[-(e_i - \mu)^2/(2\sigma^2)] \times \prod_{j=1}^{N} \exp(-a_j) \\ &\quad \times \prod_{(i,j):A_{ij}=1} \exp(a_j h_i + e_i) \Big/ \prod_{all\ i,j} (1 + \exp(a_j h_i + e_i)) \,. \end{aligned}$$

Proof: We compute that

$$\begin{aligned} &\mathbf{P}\Big(e_1 \in de_1, \ldots, e_M \in de_M, h_1 \in dh_1, \ldots, h_M \in dh_M, a_1 \in da_1, \ldots, a_N \in da_N, \{A_{ij}\}\Big) \\ &= \prod_{i=1}^{M} [\mathbf{P}(e_i \in de_i) \mathbf{P}(h_i \in dh_i)] \prod_{j=1}^{N} \mathbf{P}(a_j \in da_j) \prod_{\substack{i,j \\ A_{ij}=1}} \mathbf{P}(A_{ij} = 1 \,|\, e_i, h_i, a_j) \prod_{\substack{i,j \\ A_{ij}=0}} \mathbf{P}(A_{ij} = 0 \,|\, e_i, h_i, a_j) \\ &= \prod_{i=1}^{M} [\exp[-(e_i - \mu)^2/(2\sigma^2)] de_i \exp(-h_i) dh_i] \prod_{j=1}^{N} \exp(-a_j) da_j \prod_{\substack{i,j \\ A_{ij}=1}} \frac{\exp(a_j h_i + e_i)}{1 + \exp(a_j h_i + e_i)} \prod_{\substack{i,j \\ A_{ij}=0}} \frac{1}{1 + \exp(a_j h_i + e_i)} \\ &= \prod_{i=1}^{M} [\exp[-(e_i - \mu)^2/(2\sigma^2)] de_i \exp(-h_i) dh_i] \prod_{j=1}^{N} \exp(-a_j) da_j \prod_{\substack{i,j \\ A_{ij}=1}} \exp(a_j h_i + e_i) \Big/ \prod_{all\ i,j} (1 + \exp(a_j h_i + e_i)) \,. \end{aligned}$$

The result now follows by inspection. □

Our Bayesian algorithm then reports the conditional means of the $2M+N$ parameters, according to the posterior density π. That is, it reports final values $\widehat{a}_j$, $\widehat{h}_i$, and $\widehat{e}_i$, where, for example

$$\widehat{a}_j = \int_{\mathbf{R}^{2M+N}} a_j \pi(e_1, \ldots, e_M, h_1, \ldots, h_M, a_1, \ldots, a_N) \;\; de_1 \ldots de_M dh_1 \ldots dh_M da_1 \ldots da_N .$$

To actually compute these conditional means is non-trivial. To accomplish this, we used a *Metropolis Algorithm*. The Metropolis algorithm is an example of a *Markov chain Monte Carlo Algorithm*; for background see, e.g., Smith and Roberts [22]; Tierney [23]; Gilks et al. [11]; Roberts and Rosenthal [20].

The Metropolis Algorithm proceeds by starting all the $2M + N$ parameter values at 1. It then attempts, for each parameter in turn, to add an independent $N(0, \xi^2)$ random variable to the parameter. It then "accepts" this new value with probability $\min(1, \pi(new)/\pi(old))$, otherwise it "rejects" it and leaves the parameter value the way it is. If this algorithm is iterated enough times, and the observed parameter values at each iteration are averaged, then the resulting averages will converge (see e.g. Tierney, 1994) to the desired conditional means.

There is, of course, some arbitrariness in the specification of this Bayesian algorithm, e.g., in the form of the prior distributions and in the precise formula for the probability of a link from i to j. However, the model appears to work well in practice, as our experiments show. We note that it is possible that the priors for a new search query could instead depend on the performance of hub i on different *previous* searches, though we do not pursue that here.

This Bayesian algorithm is similar in spirit to the PHITS algorithm of Cohn and Chang [8] described earlier, in that both use statistical modeling, and both use an iterative algorithm to converge to an answer. However, the algorithms differ substantially in their details. Firstly, they use substantially different statistical models. Secondly, the PHITS algorithm uses a non-Bayesian (i.e. "classical" or "frequentist") statistical framework, as opposed to the Bayesian framework adopted here.

6.1 A Simplified Bayesian Algorithm

It is possible to simplify the above Bayesian model, by replacing equation (2) with

$$\mathbf{P}(i \to j) \;=\; \frac{a_j h_i}{1 + a_j h_i},$$

and correspondingly replace equation (3) with

$$\mathbf{P}(i \not\to j) \;=\; \frac{1}{1 + a_j h_i}.$$

This eliminates the parameters e_i entirely, so that we no longer need the prior values μ and σ. A similar model for the generation of links was considered by Azar et al. [3].

This leads to a slightly modified posterior density $\pi(\cdot)$, now given by $\pi : \mathbf{R}^{M+N} \to \mathbf{R}^{\geq 0}$ where

$$\pi(h_1, \ldots, h_M, a_1, \ldots, a_N) \;\propto \prod_{i=1}^{M} \exp(-h_i) \times \prod_{j=1}^{N} \exp(-a_j) \times \prod_{(i,j):A_{ij}=1} a_j h_i \Big/ \prod_{all\ i,j} (1 + a_j h_i).$$

This Simplified Bayesian algorithm was designed to be to similar to the original Bayesian algorithm. Surprisingly, we will see that experimentally it often performs very similarly to the SALSA algorithm.

7 Experimental Results

We have implemented the algorithms presented here on various queries. Because of space limitations we only report here (see Appendix A) a representative subset of results; all of our results (including the queries "death penalty", "computational complexity" and "gun control" which are not reported here) can be obtained at `http://www.cs.toronto.edu/~tsap/experiments`. The reader may find it easier to follow the discussion in the next section by accessing the full set of results (which includes results for the SALSA algorithm, and the Authority Threshold algorithm, when K = 1). The experimental results presented in this paper, are an improved version of the results presented in [6] where we now use an improved criterion for testing the convergence of the eigenvector algorithms.

For the generation of the Base Set of pages, we follow the specifications of Kleinberg [14] described earlier. For each of the queries, we begin by generating a Root Set that consists of the first 200 pages returned by AltaVista on the same query. The Root Set is then expanded to the Base Set by including nodes that point to, or are pointed to, by the pages in the Root Set. In order to keep the size of the Base Set manageable, for every page in the Root Set, we only include the first 50 pages returned from AltaVista that point to this page. We then construct the graph induced by nodes in the Base Set, by discovering all links among the pages in the Base Set, eliminating those that are between pages of the same domain[3].

For each query, we tested nine different algorithms on the same Base Set. We present the top ten authority sites returned by each of the algorithms. For evaluation purposes, we also include a list of the URL and title (possibly abbreviated) of each site which appears in the top five of one or more of the algorithms. For each page we also note the popularity of the page (denoted *pop* in the tables), that is, the number of different algorithms that rank it in the top ten sites. The pages that seem (to us) to be generally unrelated with the topic in hand appear bold-faced. We also present an "intersection table" which provides, for each pair of algorithms, the number of sites which were in the top ten according to *both* algorithms (maximum 10, minimum 0).

In the following we merge the SALSA and pSALSA algorithms under the name SALSA. The experiments have shown that most graphs consists of a giant component of authorities and small isolated components. Furthermore, for all the queries we performed the ranking of the first 50 pages is identical. This is not true for the full ranking; the SALSA algorithm tends to promote some pages higher than the pSALSA, because of the fact that they belong to small components. However, for the purposes of this presentation we view these two algorithms as being essentially the same.

In the tables, SBayesian denotes the Simplified Bayesian algorithm, HubAvg denotes the Hub-Averaging Kleinberg algorithm, AThresh denotes the Authority-Threshold algorithm, HThresh denotes the Hub-Threshold algorithm, and FThresh denotes the Full-Threshold algorithm. For the Authority-Threshold and Full-Threshold algorithms, we (arbitrarily) set the threshold $K = 10$.

7.1 Discussion of Experimental Results

We observe from the experiments that different algorithms emerge as the "best" for different queries, while there are queries for which no algorithm seems to perform well. One prominent such case is the query on "net censorship" (also on "computational complexity") where only a few of the top ten pages returned by any of the algorithms can possibly be considered as authoritative on the subject. One possible explanation is that in these cases the topic is not well represented on the web, or there is no strong interconnected community. This reinforces a common belief that any commercial search engine cannot rely solely on link information, but rather must also examine the text content of sites to prevent such difficulties as "topic drift". On the other hand, in cases such as "death penalty" (not shown here), all algorithms converge to almost the same top ten pages, which are both relevant and authoritative. In these cases the community is well represented, and strongly interconnected.

The experiments also indicate the difference between the behavior of the Kleinberg algorithm and SALSA, first observed in the paper of Lempel and Moran [15]. Specifically, when computing the top authorities, the Kleinberg algorithm tends to concentrate on a "tightly knit community" of nodes (the TKC effect), while SALSA, tends to mix the authorities of different communities in the top authorities. The TKC effect becomes clear in the "genetic" query (also in the "computational complexity" query), where the Kleinberg algorithm only reports pages on biology in the top ten while SALSA mixes these pages with pages on genetic algorithms. It also becomes poignantly clear in the "movies" query (and also in the "gun control" and the "abortion" query), where the top ten pages reported by the Kleinberg algorithm are dominated by an irrelevant cluster of nodes from the `about.com` community. A more elaborate algorithm for detecting intra-domain links could help alleviate this problem. However, these examples seem indicative of the topic drift potential of the principal eigenvector computed by the Kleinberg algorithm.

On the other hand, the limitations of the SALSA algorithm become obvious in the "computational geometry" query, where three out of the top ten pages belong to the unrelated `w3.com` community. They appear in the top positions because they are pointed to by a large collection of pages by ACM, which point to nothing else. A similar phenomenon explains the appearance of the "Yahoo!" page in the "genetic" query. We thus see that the simple heuristic of counting the in-degree as the authority weight is also imperfect.

[3]If one modifies the way the Base Set or the graph is constructed, the results of the algorithms can vary dramatically. In our above-mentioned web page we report the output of the algorithms for the same query, over different graphs.

We identify two types of characteristic behavior: the Kleinberg behavior, and the SALSA behavior. The former ranks the authorities based on the structure of the entire graph, and tends to favor the authorities of tightly knit communities. The latter ranks the authorities based on their popularity in their immediate neighborhood, and favors various authorities from different communities. To see how the rest of the algorithms fit within these two types of behaviors, we compare the behavior of algorithms on a pairwise basis, using the number of intersections in their respective top ten authorities as an indication of agreement.

The first striking observation is that the Simplified Bayesian algorithm is almost identical to the SALSA algorithm. The SALSA algorithm and the Simplified Bayesian have at least 80% overlap on all queries. One possible explanation for this is that both algorithms place great importance on the in-degree of a node when determining the authority weight of a node. For the SALSA algorithm we know that it is "local" in nature, that is, the authority weight assigned to a node depends only on the links that point to this node, and not on the structure of the whole graph. The Simplified Bayesian seems to possess a similar, yet weaker property; we explore the locality issue further in the next section. On the other hand, the Bayesian algorithm appears to resemble both the Kleinberg and the SALSA behavior, leaning more towards the first. Indeed, although the Bayesian algorithm avoids the severe topic drift in the "movies" and the "gun control" queries (but not in the "abortion" case), it usually has higher intersection numbers with Kleinberg than with SALSA. One possible explanation for this observation is the presence of the e_i parameters in the Bayesian algorithm (but not the Simplified Bayesian algorithm), which "absorb" some of the effect of many links pointing to a node, thus causing the authority weight of a node to be less dependent on its in-degree.

Another algorithm that seems to combine characteristics of both the SALSA and the Kleinberg behavior is the Hub-Averaging algorithm. The Hub-Averaging algorithm is by construction a hybrid of the two since it alternates between one step of each algorithm. It shares certain behavior characteristics with the Kleinberg algorithm: if we consider a full bipartite graph, then the weights of the authorities increase exponentially fast for Hub-Averaging (the rate of increase, however, is the square root of that of the Kleinberg algorithm). However, if the component becomes infiltrated, by making one of the hubs point to a node outside the component, then the weights of the authorities in the component *drop*. This prevents the Hub-Averaging algorithm from completely following the drifting behavior of the Kleinberg algorithm in the "movies" query. Nevertheless, in the "genetic" query, Hub-Averaging agrees strongly with Kleinberg, focusing on sites of a single community, instead of mixing communities as does SALSA[4]. On the other hand, Hub-Averaging and SALSA share a common characteristic, since the Hub-Averaging algorithm tends to favor nodes with high in-degree. Namely, if we consider an isolated component of one authority with high in-degree, the authority weight of this node will increase exponentially fast. This explains the fact that the top three authorities for "computational geometry" are the `w3.com` pages that are also ranked highly by SALSA (with Hub-Averaging giving a very high weight to all three authorities).

For the threshold algorithms, since they are modifications of the Kleinberg Algorithm, they are usually closer to the Kleinberg behavior. This is especially true for the Hub-Threshold algorithm. However, the benefit of eliminating unimportant hubs when computing authorities becomes obvious in the "abortion" query. The top authorities reported by the Kleinberg algorithm all belong to the `amazon.com` community, while the Hub-Threshold algorithm escapes this cluster, and produces a set of pages that are all on topic.

The Authority-Threshold often appears to be most similar with the Hub-Averaging algorithm. This makes sense since these two algorithms have a similar underlying motivation. The best moment for Authority-Threshold is the "movies" query, where it reports the most relevant top ten pages among all algorithms. An interesting case for the Authority Threshold algorithm is when we set $K = 1$. As we previously discussed, in this case the node with the highest in-degree acts as a seed to the algorithm: this node is ranked first, and the rest of the pages are ranked according to their relatedness to the seed page. Therefore, the quality of the results depends on the quality of the seed node. We present some experimental results for the case $K = 1$ in our web page. In all queries, the algorithm produces satisfactory results, except for the "net censorship" query, where the seed page is the "Yahoo" home page, so the top pages are all devoted to pages on search engines. The behavior of the algorithm is highly focused, since it only outputs pages from the community of the seed page.

The Full-Threshold algorithm combines elements of both the Threshold algorithms; however, usually it reports in the top ten a mixture of the results of the two algorithms, rather than the best of the two.

[4] In a version of the "abortion" query (denoted "refined" in our web page), the Hub-Averaging algorithm exhibits mixing of communities, similar to SALSA.

Finally, the BFS algorithm is designed to be a generalization of the SALSA algorithm, that combines some elements of the Kleinberg algorithm. Its behavior resembles both SALSA and Kleinberg, with a tendency to favor SALSA. In the "genetic" and "abortion" queries it demonstrates some mixing, but to a lesser extent than that of SALSA. The most successful moments for BFS are the "abortion" and the "gun control" queries where it reports a set of top ten pages that are all on topic. An interesting question to investigate is how the behavior of the BFS algorithm is altered if we change the weighting scheme of the neighbors.

8 Theoretical Analysis

The experimental results of the previous section suggest that certain algorithms seem to share similar properties and ranking behavior. In this section, we elaborate upon the formal study of fundamental properties and comparisons between ranking algorithms, first initiated in [6]. For the purpose of the following analysis we need some basic definitions and notation. Let $\mathcal{G}_N$ be a collection of graphs of size N. One special case is to let $\mathcal{G}_N$ be the set of *all* directed graphs of size N, hereafter denoted $\overline{\mathcal{G}}_N$. We define a link analysis algorithm A as a function that maps a graph $G \in \mathcal{G}_N$ to an N-dimensional vector. We call the vector $A(G)$ the *weight* vector of algorithm A on graph G. The value of the entry $A(G)[i]$ of vector $A(G)$ denotes the authority weight assigned by the algorithm A to the page i.

We can normalize the weight vector $A(G)$ under some chosen norm. The choice of normalization affects the definition of some of the properties of the algorithms, so we discriminate between algorithms that use different norms. For any norm L, we define the L-algorithm A to be the algorithm A, where the weight vector of A is normalized under L. For the following discussion, when not stated explicitly, we will assume that the weight vectors of the algorithms are normalized under the L_p norm for some $1 \leq p \leq \infty$.

8.1 Monotonicity

Definition 1 *An algorithm A is monotone if it has the following property: If j and k are two different nodes in a graph G, such that every hub which links to j also links to k, then $A(G)[k] \geq A(G)[j]$.*

Monotonicity appears to be a "reasonable" property but one can define "reasonable" algorithms that are not monotone. The Hub-Threshold algorithm we consider is not monotone[5]. One can find simple examples where the Hub-Threshold algorithm converges to a weight vector that does not satisfy the monotonicity property.

Theorem 2 *Except for the Hub-Threshold algorithm, all other algorithms we consider in this paper are monotone.*

Proof: Let j and k be two different nodes in a graph G, such that every node that links to j also links to k. For the pSALSA algorithm monotonicity is obvious, since the authority weights are proportional to the in-degrees of the nodes, and the in-degree of j is less than, or equal to the in-degree of k. The same holds for the SALSA algorithm within each authority connected component, which is sufficient to prove the monotonicity of the algorithm.

For the Kleinberg and Hub-Averaging algorithms, it is not hard to see that they are monotone. Indeed, regardless of the $(n-1)^{\text{th}}$ iteration hub values, the n^{th} iteration authority value for k will be at least as large as that of j, since the set of hubs that point to j is a subset of that of k. Hence, for every $n \geq 1$, the monotonicity property holds at the end of the n^{th} iteration; therefore, the algorithms are monotone as $n \to \infty$. Proofs of monotonicity for the Authority-Threshold Kleinberg algorithms and the BFS algorithm follow similarly.

For the Bayesian algorithm the proof of monotonicity is more involved. Recall that the Bayesian algorithm leads to a density of the form

$$\pi(e_1, \ldots, e_M, h_1, \ldots, h_M, a_1, \ldots, a_N) \;\propto\; (\text{prior density}) \times \prod_{(i,j):A_{ij}=1} \exp(a_j h_i + e_i) \;\;/\;\; \prod_{(i,j)} (1 + \exp(a_j h_i + e_i)) \,.$$

[5] There are many variations of the Hub-Threshold algorithm that are monotone. For example, if we set the threshold to be the median hub value (or some fixed value) instead of the mean, then the algorithm is monotone.

Now, let j and k be two different authority pages, such that every hub which links to j also links to k. Consider the conditional distribution of a_j and a_k, conditional on the values of $e_1, \ldots, e_M, h_1, \ldots, h_M$. We see that

$$\pi(a_j \mid e_1, \ldots, e_M, h_1, \ldots, h_M) \propto (\text{prior density}) \times \prod_{i:A_{ij}=1} \exp(a_j h_i) \;/\; \prod_i (1 + \exp(a_j h_i + e_i)),$$

and

$$\pi(a_k \mid e_1, \ldots, e_M, h_1, \ldots, h_M) \propto (\text{prior density}) \times \prod_{i:A_{ik}=1} \exp(a_k h_i) \;/\; \prod_i (1 + \exp(a_k h_i + e_i)).$$

Hence,

$$\frac{\pi(a_k \in da \mid e_1, \ldots, e_M, h_1, \ldots, h_M)}{\pi(a_j \in da \mid e_1, \ldots, e_M, h_1, \ldots, h_M)} \propto \exp\left(a \sum_{i:A_{ik}=1, A_{ij}=0} h_i \right), \tag{5}$$

regardless of the choice of prior density.

Now, since $h_i \geq 0$ for all i, it is seen that the expression (5) is a non-decreasing function of a. It then follows from the well-known "FKG inequality" (see e.g. Lemma 5 of Roberts and Rosenthal [21]) that, in the distribution π,

$$\mathbf{P}(a_k \geq a \mid e_1, \ldots, e_M, h_1, \ldots, h_M) \geq \mathbf{P}(a_j \geq a \mid e_1, \ldots, e_M, h_1, \ldots, h_M), \qquad a \in \mathbf{R}, \tag{6}$$

i.e., that the conditional distribution of a_j stochastically dominates that of a_i. But then, integrating both sides of the inequality (6) over the joint distribution of $(e_1, \ldots, e_M, h_1, \ldots, h_M)$, it follows that in the distribution π,

$$\mathbf{P}(a_k \geq a) \geq \mathbf{P}(a_j \geq a), \qquad a \in \mathbf{R},$$

i.e., that the unconditional distribution of a_k stochastically dominates that of a_j. In particular, the mean of a_k under the posterior distribution π is at least as large as that of a_j. Hence, the Bayesian Algorithm gives a higher authority weight to authority k than to authority j, completing the proof of monotonicity of the Bayesian algorithm.

For the Simplified Bayesian algorithm, the argument is similar. In this case, the expression (5) is replaced by

$$\frac{\pi(a_k \in da \mid e_1, \ldots, e_M, h_1, \ldots, h_M)}{\pi(a_j \in da \mid e_1, \ldots, e_M, h_1, \ldots, h_M)} \propto \prod_{i:A_{ik}=1, A_{ij}=0} a h_i.$$

Since $h_i \geq 0$, we again see that this is a non-decreasing function of a, and a similar proof applies. □

8.2 Similarity

Let A_1 and A_2 be two algorithms on $\mathcal{G}_N$. We consider the *distance* $d(A_1(G), A_2(G))$ between the weight vectors of $A_1(G)$ and $A_2(G)$, for $G \in \mathcal{G}_N$, where $d : \mathbf{R}^n \times \mathbf{R}^n \to \mathbf{R}$ is some function that maps the weight vectors w_1 and w_2 to a real number $d(w_1, w_2)$. We first consider the Manhattan distance d_1, that is, the L_1 norm of the difference of the weight vectors, given by $d_1(w_1, w_2) = \sum_{i=1}^N |w_1(i) - w_2(i)|$.

For this distance function, we now define the similarity between two L_p-algorithms as follows. For the following, if γ is a constant, and w is a vector, then γw denotes the usual scalar-vector product.

Definition 2 *Let $1 \leq p \leq \infty$. Two L_p-algorithms A_1 and A_2 are similar on $\mathcal{G}_N$, if (as $N \to \infty$)*

$$\max_{G \in \mathcal{G}_N} \min_{\gamma_1, \gamma_2 \geq 1} d_1(\gamma_1 A_1(G), \gamma_2 A_2(G)) = o(N^{1-1/p}).$$

The choice for the bound $o(N^{1-1/p})$ in the definition of similarity is guided by the fact that the maximum d_1 distance between any two N-dimensional L_p unit vectors is $\Theta(N^{1-1/p})$. This definition of similarity generalizes the definition in [6], where $\gamma_1 = \gamma_2 = 1$. The constants γ_1 and γ_2 are introduced so as to allow for an arbitrary scaling of the two vectors, thus eliminating dissimilarity that is caused solely due to normalization factors. For example, let $w_1 = (1, 1, ..., 1, 2)$, and $w_2 = (1, 1, ..., 1)$ be two weight vectors before any normalization is applied. These two vectors appear to be similar. However, if we normalize in the L_∞ norm, then for $\gamma_1 = \gamma_2 = 1$, $d_1(w_1, w_2) = \Theta(N)$; therefore, in the original definition, the vectors would be dissimilar.

We also consider another distance function that attempts to capture the similarity between the *ordinal* rankings of two algorithms. The motivation behind this definition is that the ordinal ranking is the usual end-product seen by the user. Let $w_1 = A_1(G)$ and $w_2 = A_2(G)$ be the weight vectors of two algorithms A_1 and A_2. We define the indicator function $\mathbf{I}_{w_1 w_2}(i, j)$ as follows

$$\mathbf{I}_{w_1 w_2}(i, j) = \begin{cases} 1 & \text{if } w_1(i) < w_1(j) \text{ AND } w_2(i) > w_2(j) \\ 0 & \text{otherwise} \end{cases}$$

We note that $\mathbf{I}_{w_1 w_2}(i, j) = 0$ if and only if $w_1(i) < w_2(j) \Rightarrow w_2(i) \leq w_2(j)$. $\mathbf{I}_{w_1 w_2}(i, j)$ becomes one for each pair of nodes that are ranked differently. We define the "ranking distance" function d_r as follows.

$$d_r(w_1, w_2) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} \mathbf{I}_{w_1 w_2}(i, j) \ .$$

Note that, unlike d_1, the distance between two weight vectors under d_r does not depend upon the choice of normalization. Similar distance measures between rankings are examined by Dwork et al. [10].

Definition 3 *Two algorithms, A_1 and A_2, are rank similar on $\mathcal{G}_N$, if (as $N \to \infty$)*

$$\max_{G \in \mathcal{G}_N} \ d_r(A_1(G), A_2(G)) \ = \ o(N) \ .$$

Definition 4 *Two algorithms, A_1 and A_2, are rank matching on $\{\mathcal{G}_N\}$, if for every graph $G \in \mathcal{G}_N$,*

$$d_r(A_1(G), A_2(G)) = 0 \ .$$

Remark: We note that by the above definition, every algorithm is rank matching with the trivial algorithm that gives the same weight to all authorities. Although this may seem somewhat bizarre, it does have an intuitive justification. For an algorithm whose goal is to produce an *ordinal* ranking, the weight vector with all weights equal conveys no information; therefore, it lends itself to all possible ordinal rankings. We also note that the d_r distance does not satisfy the triangle inequality, since, e.g., all algorithms have d_r-distance 0 to the trivial algorithm. Of course, it is straightforward to modify the definition of d_r to avoid this; however, we find the definition used here to be most natural.

For the purposes of this paper, we only consider the d_1 and d_r distance functions. Nevertheless, the definition of similarity can be generalized to any distance function d, and any normalization norm $||\cdot||$, as follows.

Definition 5 *Two L-algorithms A_1 and A_2 are similar on $\mathcal{G}_N$ under d, if (as $N \to \infty$)*

$$\max_{G \in \mathcal{G}_N} \ \min_{\gamma_1, \gamma_2 \geq 1} \ d(\gamma_1 A_1(G), \gamma_2 A_2(G)) = o(M_N) \ ^6,$$

where $M_N = \sup_{||w_1|| = ||w_2|| = 1} d(w_1, w_2)$ is the maximum distance between any two N-vectors with unit norm $L = ||\cdot||$.

The definition of similarity depends on the normalization of the algorithms. In the following we show that for the d_1 distance, similarity in L_p norm implies similarity in L_q norm, for any $q > p$.

Lemma 2 *Let v be a vector of length N, and suppose $1 \leq r < s \leq \infty$. Then $||v||_r \leq ||v||_s N^{1/r - 1/s}$.*

[6] Other operators are also possible. For example, if there exists some distribution over the graphs in $\mathcal{G}_N$ we could replace "max" by the average distance between the algorithms.

Proof: Assume first that $s < \infty$. We use Hölder's inequality, which states that for any p and q such that $1 < p, q < \infty$ and $1/p + 1/q = 1$, if x and y are two N-dimensional vectors, then

$$\sum_{i=1}^{N} |x(i)y(i)| \leq \left(\sum_{i=1}^{N} |x(i)|^p\right)^{1/p} \left(\sum_{i=1}^{N} |y(i)|^q\right)^{1/q} .$$

Set $p = s/r$ and $q = 1/(1 - 1/p)$. Also, set $x(i) = v(i)^r$ and $y(i) \equiv 1$, and let $\|v\|_r^r$ denote $(\|v\|_r)^r$, and $\|v\|_s^r$ denote $(\|v\|_s)^r$. We have that

$$\begin{aligned} \|v\|_r^r &= \sum_{i=1}^{N} |v(i)|^r = \sum_{i=1}^{N} |v(i)|^r \, 1 \leq \left(\sum_{i=1}^{N} |v(i)|^{r(s/r)}\right)^{1/(s/r)} \left(\sum_{i=1}^{N} 1^q\right)^{1/q} \\ &= \|v\|_s^r N^{1/q} = \|v\|_s^r N^{1-1/(s/r)} = \|v\|_s^r N^{1-r/s} . \end{aligned}$$

Taking r^{th} roots of both sides, we obtain $\|v\|_r \leq \|v\|_s N^{1/r-1/s}$, as claimed.

For the case $s = \infty$, we compute that

$$\|v\|_r^r = \sum_{i=1}^{N} |v(i)|^r \leq \sum_{i=1}^{N} \max_i |v(i)|^r = N \max_i |v(i)|^r = N\|v\|_\infty^r .$$

Thus, $\|v\|_r \leq N^{1/r}\|v\|_\infty$. □

Theorem 3 *Let A_1 and A_2 be two algorithms, and let $1 \leq r \leq s \leq \infty$. If the L_r-algorithm A_1 and the L_r-algorithm A_2 are similar, then the L_s-algorithm A_1 and the L_s-algorithm A_2 are also similar.*

Proof: Let G be a graph of size N, and let $u = A_1(G)$, and $v = A_2(G)$ be the weight vectors of the two algorithms. Let v_p and u_p denote the weight vectors, normalized in the L_p norm. Since the L_r-algorithm A_1 and the L_r-algorithm A_2 are similar, there exist $\gamma_1, \gamma_2 \geq 1$ such that

$$d_1(\gamma_1 v_r, \gamma_2 u_r) = \sum_{i=1}^{N} |\gamma_1 v_r(i) - \gamma_2 u_r(i)| = o(N^{1-1/r}) .$$

Now, $v_s = v_r/\|v_r\|_s$, and $u_s = u_r/\|u_r\|_s$. Therefore, $\sum_{i=1}^{N} |\gamma_1 \|v_r\|_s v_s(i) - \gamma_2 \|u_r\|_s u_s(i)| = o(N^{1-1/r})$. Without loss of generality assume that $\|u_r\|_s \geq \|v_r\|_s$. Then

$$\|v_r\|_s \sum_{i=1}^{N} \left|\gamma_1 v_s(i) - \gamma_2 \frac{\|u_r\|_s}{\|v_r\|_s} u_s(i)\right| = o(N^{1-1/r}) .$$

We set $\gamma_1' = \gamma_1$ and $\gamma_2' = \gamma_2 \frac{\|u_r\|_s}{\|v_r\|_s}$. Then we have that

$$d_1(\gamma_1' v_s, \gamma_2' u_s) = \sum_{i=1}^{N} |\gamma_1' v_s(i) - \gamma_2' u_s(i)| = o\left(\frac{N^{1-1/r}}{\|v_r\|_s}\right) .$$

But from the lemma, $\|v_r\|_s \geq \|v_r\|_r N^{1/s-1/r} = N^{1/s-1/r}$. Hence, $\frac{N^{1-1/r}}{\|v_r\|_s} \leq \frac{N^{1-1/r}}{N^{1/s-1/r}} = N^{1-1/s}$. Therefore, $d_1(\gamma_1' v_s, \gamma_2' u_s) = o(N^{1-1/s})$, and thus L_s-algorithm A_1, and L_s-algorithm A_2 are similar. □

Theorem 3 implies that if two L_1-algorithms are similar, then the corresponding L_p-algorithms are also similar, for any $1 \leq p \leq \infty$. Furthermore, if two L_∞-algorithms are dissimilar, then the corresponding L_p-algorithms are also dissimilar, for any $1 \leq p \leq \infty$. Therefore, the following dissimilarity results, proven for the L_∞ norm, hold for any L_p norm, for $1 \leq p \leq \infty$.

Proposition 1 *The Hub-Averaging algorithm, and the Kleinberg algorithm are neither similar, nor rank similar on* $\overline{\mathcal{G}}_N$.

Proof: Consider a graph G on $N = 3r$ nodes that consists of two disconnected components. The first component C_1 consists of a complete graph on r nodes. The second component C_2 consists of a complete graph C on r nodes, and a set of r "external" nodes E, such that each node in C points to a node in E, and no two nodes in C point to the same "external" node.

Let w_K and w_H denote the weight vectors of the Kleinberg, and the Hub-Averaging algorithm, respectively, on graph G. We assume that the vectors are normalized in L_∞ norm. It is not hard to see that the Kleinberg algorithm allocates all the weight to the nodes in C_2. After normalization, for all $i \in C$, $w_K(i) = 1$, for all $j \in E$, $w_K(j) = \frac{1}{r-1}$, and for all $k \in C_1$, $w_K(k) = 0$. On the other hand, the Hub-Averaging algorithm allocates all the weight to the nodes in C_1. After normalization, for all $k \in C_1$, $w_H(k) = 1$, and for all $j \in C_2$, $w_H(j) = 0$.

Let $U = C_1 \cup C$. The set U contains $2r$ nodes. For every node $i \in U$, either $w_K(i) = 1$ and $w_H(i) = 0$, or $w_K(i) = 0$ and $w_H(i) = 1$. Therefore, for all $\gamma_1, \gamma_2 \geq 1$, $\sum_{i \in U} |\gamma_1 w_K(i) - \gamma_2 w_H(i)| \geq 2r$. Thus, $d_1(\gamma_1 w_K, \gamma_2 w_H) = \Omega(r) = \Omega(N)$ which proves that the algorithms are not similar.

The proof for rank dissimilarity follows immediately from the above. For every pair of nodes (i, j) such that $i \in C_1$ and $j \in C_2$, $w_K(i) < w_K(j)$, and $w_H(i) > w_H(j)$. There are $\Theta(N^2)$ such pairs, therefore, $d_r(w_K, w_H) = \Theta(N)$. Thus, the two algorithms are not rank similar. $\square$

Proposition 2 *The pSALSA algorithm and the Hub-Averaging algorithm are neither similar, nor rank similar on* $\overline{\mathcal{G}}_N$.

Proof: For the proof of dissimilarity, we consider the same graph as in the proof of Proposition 1 for the dissimilarity between the Hub-Averaging and the Kleinberg algorithm. Let w_p and w_H be the weight vectors of pSALSA and Hub-Averaging, respectively. We assume that the vectors are normalized in L_∞ norm. For this graph, the pSALSA algorithm allocates weight $w(i) = 1$ to all nodes in C_1.On the other hand the Hub-Averaging algorithm allocates weight 1 to the nodes in C_1. There are r nodes in C_1 for which $w_p(i) = 1$ and $w_K(i) = 0$. For all $\gamma_1, \gamma_2 \geq 1$, $\sum_{i \in C_1} |\gamma_1 w_p(i) - \gamma_2 w_k(i)| \geq r$. Therefore, $d_1(\gamma_1 w_p, \gamma_2 w_K) = \Omega(r) = \Omega(N)$ which proves that Hub-Averaging and pSALSA are not similar.

For the proof of rank dissimilarity, we consider a graph G on $N = 3r + 3$ nodes which are connected as follows. The graph consists of two sets of hubs X and Y of size r and 2, respectively, and two sets of authorities A and B, each of size r, and a single "central" authority c. Each hub in set X points to exactly one distinct authority in A, and both hubs in Y point to all authorities in B. Furthermore, all hubs in X and Y point to c.

The pSALSA algorithm allocates the most weight to the central authority, then to the authorities in B, and then to the authorities in A. On the other hand, the Hub-Averaging algorithm considers each hub in X to be much better than each hub in Y. Hence, it will allocate highest weight to the authority c, nearly as high weight to the authorities in A, and much lower weight to the authorities in B. The sets A and B have size $\Theta(N)$. Therefore, there are $\Theta(N^2)$ pairs of nodes that are ranked differently from the two algorithms. Hence, Hub-Averaging and pSALSA are not rank similar. $\square$

Proposition 3 *The pSALSA algorithm and the Kleinberg algorithm are neither similar, nor rank similar on* $\overline{\mathcal{G}}_N$.

Proof: Consider a graph G on $N = 4r$ nodes that consists of two disconnected components. The first component C_1 consists of a complete graph on r nodes. The second component C_2 consists of a bipartite graph with $2r$ hubs, and r authorities. Without loss of generality assume that r is even, and enumerate all hubs and authorities in C_2. Make all "odd" hubs point to all "odd" authorities, and all "even" hubs point to all "even" authorities. Thus, each hub points to $\frac{r}{2}$ authorities, and each authority is pointed to by r authorities.

Let w_K and w_p denote the weight vectors of the Kleinberg, and the pSALSA algorithm, respectively, on graph G. We assume that the vectors are normalized in L_∞ norm. It is not hard to see that the Kleinberg algorithm allocates all the weight to the nodes in C_1. After normalization, for all $i \in C_1$, $w_K(i) = 1$, while for all $j \in C_2$, $w_K(j) = 0$. On the other hand, the pSALSA algorithm distributes the weight to both components, allocating more weight to the nodes in C_2. After the normalization step, for all $j \in C_2$, $w_p(j) = 1$, while for all $i \in C_1$, $w_p(i) = \frac{r-1}{r}$.

There are r nodes in C_2 for which $w_p(i) = 1$ and $w_K(i) = 0$. For all $\gamma_1, \gamma_2 \geq 1$, $\sum_{i \in C_2} |\gamma_1 w_p(i) - \gamma_2 w_k(i)| \geq r$. Therefore, $d_1(\gamma_1 w_p, \gamma_2 w_K) = \Omega(r) = \Omega(N)$ which proves that the algorithms are not similar.

The proof for rank dissimilarity follows immediately from the above. For every pair of nodes (i,j) such that $i \in C_1$ and $j \in C_2$, $w_K(i) > w_K(j)$, and $w_p(i) < w_p(j)$. There are $\Theta(N^2)$ such pairs, therefore, $d_r(w_K, w_H) = \Theta(N)$. Thus, the two algorithms are not rank similar. □

Proposition 4 *The SALSA algorithm is neither similar, nor rank similar with the pSALSA, Hub-Averaging, or Kleinberg algorithm.*

Proof: Consider a graph G on $N = 3r$ nodes, that consists of two components C_1 and C_2. The component C_1 is a complete graph on $2r$ nodes, and the component C_2 is a complete graph on r nodes, with one link (q,p) removed.

Let w_S, w_K, w_H, and w_p denote the weight vectors for SALSA, Kleinberg, Hub-Averaging and pSALSA algorithms respectively. We assume that the vectors are normalized in L_∞ norm. Also, let u_S denote the SALSA weight vector before normalization. The SALSA algorithm allocates weight $u_S(i) = 1/3r$ for all $i \in C_1$, and weight $u_S(j) = (r-1)/3(r^2 - r - 1)$ for all $j \in C_2 \setminus \{p\}$. It is interesting to note that the removal of the link (q,p) increased the weight of the rest of the nodes in C_2. Therefore, after normalization $w_S(i) = 1 - \frac{1}{r(r-1)}$ for all $i \in C_1$, and $w_S(j) = 1$ for all $j \in C_2 \setminus \{p\}$. On the other hand, both the Kleinberg and Hub-Averaging algorithms distribute all the weight equally to the authorities in the C_1 component, and allocate zero weight to the nodes in the C_2 component. Therefore, after normalization, $w_K(i) = w_H(i) = 1$ for all nodes $i \in C_1$, and $w_K(j) = w_H(j) = 0$ for all nodes $j \in C_2$. The pSALSA algorithm allocates weight proportionally to the in-degree of the nodes, therefore, after normalization, $w_p(i) = 1$ for all nodes in C_1, while $w_p(j) = \frac{r-1}{2r-1}$ for all nodes $j \in C_2 \setminus \{p\}$.

For the Kleinberg and Hub-Averaging algorithm, there are $r-1$ entries in $C_2 \setminus \{p\}$, for which $w_K(i) = w_H(i) = 0$ and $w_S(i) = 1$. Therefore, for all of $\gamma_1, \gamma_2 \geq 1$, $d_1(\gamma_1 w_S(i), \gamma_2 w_K(i)) = \Omega(r) = \Omega(N)$, and $d_1(\gamma_1 w_S(i), \gamma_2 w_H(i)) = \Omega(r) = \Omega(N)$. From the above, it is easy to see that $d_r(w_S, w_K) = \Theta(N)$, and $d_r(w_S, w_H) = \Theta(N)$.

The proof for the pSALSA algorithm, is a little more involved. Let

$$
\begin{aligned}
S_1 &= \sum_{i \in C_1} |\gamma_1 w_p(i) - \gamma_2 w_S(i)| = r \left| \gamma_1 - \gamma_2 - \frac{\gamma_2}{r(r-1)} \right| \\
S_2 &= \sum_{i \in C_2 \setminus \{p\}} |\gamma_1 w_p(i) - \gamma_2 w_S(i)| = (r-1) \left| \gamma_1 \frac{r-1}{2r-1} - \gamma_2 \right| .
\end{aligned}
$$

We have that $d_1(\gamma_1 w_p, \gamma_2 w_S) \geq S_1 + S_2$. It is not hard to see that unless $\gamma_1 = 2\gamma_2 + o(1)$, then $S_2 = \Theta(r) = \Theta(N)$. If $\gamma_1 = 2\gamma_2 + o(1)$ then $S_1 = \Theta(r) = \Theta(N)$. Therefore, for all $\gamma_1, \gamma_2 \geq 1$, $d_1(\gamma_1 w_p, \gamma_2 w_S) = \Omega(N)$. From the above it is easy to see that $d_r(w_S, w_p) = \Theta(N)$.

Thus, SALSA is neither similar, nor rank similar with any of the other algorithms. □

On the positive side we have the following.

Definition 6 *A link graph is "nested" if for every pair of nodes j and k, the set of in-links to j is either a subset or a superset of the set of in-links to k.*

Let $\mathcal{G}_N^{nest}$ be the set of all size-N nested graphs. (Of course, $\mathcal{G}_N^{nest}$ is a rather restricted set of size-N graphs.)

Theorem 4 *If two algorithms are both monotone, then they are rank matching on $\mathcal{G}_N^{nest}$.*

Proof: Let G be a graph in $\mathcal{G}_N^{nest}$, and let A_1 and A_2 be two monotone algorithms. Let $w_1 = A_1(G)$, and $w_2 = A_2(G)$ be the weight vectors of A_1 and A_2 respectively. Consider a pair j and k of nodes in G. Without loss of generality, assume that the set of in-links of j is a superset of the set of in-links of k. Since both algorithms are monotone, then $w_1(j) \geq w_1(k)$, and $w_2(j) \geq w_2(k)$. Therefore $\mathbf{I}_{w_1 w_2}(j,k) = 0$ for all pairs of nodes. Therefore, $d_r(w_1, w_2) = 0$, so the algorithms A_1 and A_2 are rank matching on $\mathcal{G}_N^{nest}$. □

Corollary 1 *Except for the Hub-Threshold algorithm, all other algorithms we consider in this paper are rank matching on $\mathcal{G}_N^{nest}$.*

8.3 Stability and Locality

In the previous section we examined the similarity of two different algorithms on the same graph G. In this section we are interested in how the output of a *fixed* algorithm changes, as we alter the graph. We would like small changes in the graph to have a small effect on the weight vector of the algorithm. We capture this requirement by the definition of stability. The notion of stability has been independently considered (but not explicitly defined) in a number of different papers [17, 18, 3, 1].

Given a graph G, we can view a *change* in graph G, as an operation ∂ on graph G, that adds and/or removes links so a to produce a new graph $G' = \partial G$. Formally, a change is defined as an operation on the adjacency matrix of the graph G, that alters k entries of the matrix, for some $k > 0$. The number k is called the *size* of the change. We denote by $\mathcal{C}_k$ the set of all possible changes of size at most k. We think of a change in graph G as being *small*, if the size k of the change is constant and independent of the size of the graph G.

For the following, let $E(G)$ denote the set of all edges (i.e. links) in the graph G. We assume that $E(G) = \omega(1)$, otherwise all properties that we discuss below are trivial. The following definition applies to L_p-algorithms, for $1 \leq p \leq \infty$.

Definition 7 *An L_p-algorithm A is stable[7] on $\mathcal{G}_N$ if for every fixed positive integer k, we have (as $N \to \infty$)*

$$\max_{G \in \mathcal{G}_N, \partial \in \mathcal{C}_k} \min_{\gamma_1, \gamma_2 \geq 1} d_1(\gamma_1 A(G), \gamma_2 A(\partial G)) = o(N^{1-1/p}) \ .$$

Again, our choice for the bound $o(N^{1-1/p})$ is is guided by the fact that the maximum d_1 distance between any two N-dimensional L_p unit vectors, is $\Theta(N^{1-1/p})$. As in the definition of similarity, the parameters γ_1, γ_2 used in the definition of stability allow for an arbitrary scaling of the weight vectors, thus eliminating instability which is caused solely by different normalization factors.

Definition 8 *An algorithm A is rank stable on $\mathcal{G}_N$ if for every k, we have (as $N \to \infty$)*

$$\max_{G \in \mathcal{G}_N, \partial G \in \mathcal{C}_k} d_r(A(G), A(\partial G)) \ = \ o(N).$$

As in the case of similarity, the notion of stability can be defined for any distance function, and for any normalization norm.

Definition 9 *An L-algorithm A is stable on $\mathcal{G}_N$ under d if for every fixed positive integer k, we have (as $N \to \infty$)*

$$\max_{G \in \mathcal{G}_N, \partial G \in \mathcal{C}_k} \min_{\gamma_1, \gamma_2 \geq 1} d(\gamma_1 A(G), \gamma_2 A(\partial G)) = o(M_N) \,,$$

where again $M_N = \sup_{\|w_1\| = \|w_2\| = 1} d(w_1, w_2)$ is the maximum distance between any two N-vectors with unit norm $L = \|\cdot\|$.

Stability may be a desirable property. Indeed, the algorithms all act on a base set which is generated using some other search engine (e.g. AltaVista [2]) and the associated hypertext links. Presumably with a "very good" base set, all the algorithms would perform well. However, if an algorithm is not stable, then slight changes in the base set (or its link structure) may lead to large changes in the rankings given by the algorithm. Thus, stability may provide "protection" from poor base sets.

Theorem 5 *Let A be an algorithm, and let $1 \leq r \leq s \leq \infty$. If the L_r-algorithm A is stable, then the L_s-algorithm A is also stable.*

Proof: Let G be a graph, and let ∂ denote a change of constant size in graph G. Set $v = A(G)$, and $u = A(\partial G)$, and then the rest of the proof is identical to the proof of Theorem 3. □

Theorem 5 implies that if an L_1-algorithm A is stable, then the L_p-algorithm A is also stable, for any $1 \leq p \leq \infty$. Furthermore, if the L_∞-algorithm A is unstable, then the L_p-algorithm A is also unstable, for any $1 \leq p \leq \infty$. Therefore, the following instability results, proven for the L_∞ norm, hold for any L_p norm, for $1 \leq p \leq \infty$.

[7] This definition of stability generalizes the definition in [6], where we considered only changes that remove a constant number of links.

Proposition 5 *The Kleinberg and Hub-Averaging algorithms are neither stable, nor rank stable.*

Proof: Consider the graph G of size $N = 2r + 3$, which consists of two disjoint components C_1 and C_2. The component C_1 consists of a complete graph on r nodes, and two extra hubs p and q that each points to a single node of the complete graph. The component C_2 consists of a complete graph on r nodes, and an extra node that points to exactly one of these r nodes. For both Kleinberg and Hub-Averaging transition matrices, the eigenvalue of the component C_1 is (slightly) larger than that of C_2; therefore, both algorithms will allocate all the weight to the nodes of C_1, and zero weight to C_2. If we delete the links from p and q, the eigenvalue of C_2 becomes larger, causing the all the weight to shift from C_1 to C_2, and leaving the nodes in C_1 with zero weight. It follows that the two algorithms are neither stable nor rank stable. □

Proposition 6 *The SALSA algorithm, is neither stable, nor rank stable*

Proof: We first establish the rank instability of the SALSA algorithm. The example is similar to that used in the previous proof. Consider a graph G of size $N = 2r + 3$, which consists of two disjoint components. The first component consists of a complete graph C_1 on r nodes and two extra authorities p and q each of which is pointed to by a single node of the complete graph. The second component consists of a complete graph C_2 on r nodes and an extra authority that is pointed to by exactly one of these r nodes.

It is not hard to show that if $r > 2$, then the SALSA algorithm ranks the r authorities in C_1 higher than those in C_2. We now remove the links to the nodes p and q. Simple computations show that if $r > 1$, the nodes in C_2 are ranked higher than the nodes in C_1. There are $\Theta(N^2)$ pairs of nodes whose relative order is changed; therefore, SALSA is rank unstable.

The proof of instability is a little more involved. Consider again the graph G that consists of two complete graphs C_1 and C_2 of size N_1 and N_2 respectively, such that $N_2 = cN_1$, where $c < 1$ is a fixed constant. There exists also an extra hub h that points to two authorities p and q from the components C_1 and C_2 respectively. The graph has $N = N_1 + N_2 + 1$ nodes, and $N_A = N_1 + N_2$ authorities.

The authority Markov chain defined by the SALSA algorithm is irreducible, therefore, the weight of authority i is proportional to the in-degree of node i. Let w be the weight vector of the SALSA algorithm. Node p is the node with the highest in-degree, therefore, after normalizing in the L_∞ norm, $w(p) = 1$, $w(i) = 1 - 1/N_1$, for all $i \in C_1 \setminus \{p\}$, $w(q) = c$, and $w(j) = c - 1/N_1$ for all $j \in C_2 \setminus \{q\}$.

Now let G' be the graph G after we remove the two links from hub h to authorities p and q. Let w' denote the weight vector of the SALSA algorithm on graph G'. It is not hard to see that all authorities receive the same weight $1/N_A$ by the SALSA algorithm. After normalization, $w'(i) = 1$ for all authorities i in G'.

Consider now the distance $d_1(\gamma_1 w, \gamma_2 w')$. Let

$$\begin{aligned} S_1 &= \sum_{C_1\setminus\{p\}} |\gamma_1 w(i) - \gamma_2 w'(i)| = (N_1 - 1)\left|\gamma_1 - \gamma_2 - \frac{\gamma_1}{N_1}\right| \\ S_2 &= \sum_{C_2\setminus\{q\}} |\gamma_1 w(i) - \gamma_2 w'(i)| = (N_2 - 1)\left|c\gamma_1 - \gamma_2 - \frac{\gamma_1}{N_1}\right| . \end{aligned}$$

It holds that $d_1(\gamma_1 w, \gamma_2 w') \geq S_1 + S_2$. It is not hard to see that unless $\gamma_1 = \frac{1}{c}\gamma_2 + o(1)$, then $S_2 = \Theta(N_2) = \Theta(N)$. If $\gamma_1 = \frac{1}{c}\gamma_2 + o(1)$, then $S_1 = \Theta(N_1) = \Theta(N)$. Therefore, $d_1(\gamma_1 w, \gamma_2 w') = \Omega(N)$. Thus, the SALSA algorithm is unstable. □

We now introduce the idea of "locality". The idea behind locality is that a change in the in-links of a node should have only a small effect on the weights of the rest of the nodes. Given a graph G, we say that a change ∂ in G *affects* node i, if the links that point to node i are altered. In algebraic terms, the change ∂ affects the entries of the i^{th} column of the adjacency matrix of graph G. We define the *impact set* of a change ∂ in graph G, $\{\partial G\}$, to be the set of nodes in G affected by the change ∂.

Definition 10 *An algorithm A is local*[8] *if for every graph G, and every change ∂ in G there exists $\lambda > 0$ such that $A(\partial G)[i] = \lambda A(G)[i]$, for all $i \notin \{\partial G\}$.*

Definition 11 *An algorithm A is* rank local *if for every graph G, and every change ∂ in G, if $w = A(G)$ and $w' = A(\partial G)$, then, for all $i, j \notin \{\partial G\}$, $w(i) > w(j) \Rightarrow w'(i) \geq w'(j)$ (equivalently, $\mathbf{I}_{ww'}(i,j) = 0$). The algorithm is* strictly rank local[9] *if $w(i) > w(j) \Leftrightarrow w'(i) > w'(j)$.*

We note that locality and rank locality do not depend upon the normalization used by the algorithm. From the definitions, one can observe that if an algorithm is local, then it is also strictly rank local. If it is strictly rank local then it is obviously rank local.

We have the following.

Theorem 6 *If an algorithm is rank local, then it is rank stable.*

Proof: Let G be a graph, and let ∂ be a change of size k in G. Let w be the weight vector of the algorithm on a graph G, and let w' be the weight vector of the algorithm on the modified graph ∂G. Let $P = \{\partial G\}$ be the impact set of change ∂, and let m be the size of the set P. Since the algorithm is rank local, $\mathbf{I}_{ww'}(i,j) = 0$ for all $i, j \notin P$. Therefore,

$$d_r(w, w') = \frac{1}{N} \sum_{i=1}^{N} \sum_{p \in P}^{m} \mathbf{I}_{ww'}(i,p) \ .$$

But $\mathbf{I}_{ww'}(i,p) \leq 1$ for all i and p, so $d_r(w,w') = \frac{1}{N} Nm = m$. Therefore, the algorithm is rank stable. □

Therefore, locality implies rank stability. It is not necessarily the case that it also implies stability. For example, consider the algorithm A where for a graph G on N nodes, it assigns weight $N^{|B(i)|}$ to node i. This algorithm is local, but it is not stable.

Theorem 7 *The pSALSA algorithm is local, and consequently strictly rank local, and rank local.*

Proof: Given a graph G, let u be the weight vector that assigns to node i weight equal to $|B(i)|$, the in-degree of i. Let w be the weight vector of the pSALSA algorithm; then $w(i) = u(i)/\|u\| = |B(i)|/\|u\|$, where $\|\cdot\|$ is any norm.

Let ∂ be a change in G, and let $G' = \partial G$ denote the new graph. Let u' and w' denote the corresponding weight vectors on graph G'. For every $i \notin \{\partial G\}$, the number of links to i remains unaffected by the change ∂; therefore $u'(i) = u(i)$. For the pSALSA algorithm, $w'(i) = u'(i)/\|u'\| = u(i)/\|u'\|$. For $\lambda = \frac{\|u\|}{\|u'\|}$, it holds that $w'(i) = \lambda w(i)$, for all $i \notin \{\partial G\}$. Thus, pSALSA is local, and consequently strictly rank local, and rank local. □

Theorem 8 *The pSALSA algorithm is stable, and rank stable.*

Proof: The proof of rank stability follows directly from the rank locality of pSALSA. For the proof of stability, let G be a graph on N nodes, and let $\partial \in \mathcal{C}_k$ be a change of size k in G. Let m be the size of $\{\partial G\}$. Without loss of generality assume that $\{\partial G\} = \{1, 2, \ldots, m\}$. Let u be the weight vector that assigns to node i weight equal to $|B(i)|$, the in-degree of i. Let w be the weight of the L_1-pSALSA algorithm. Then $w = u/\|u\|$, where $\|\cdot\|$ is the L_1 norm. Let u' and w' denote the corresponding weight vectors after the removal of the links. For all $i \notin \{1, 2, \ldots, m\}$ $u'(i) = u(i)$. Furthermore, $\sum_{i=1}^{m} |u(i) - u'(i)| \leq k$. Set $\gamma_1 = 1$ and $\gamma_2 = \frac{\|u'\|}{\|u\|}$. Then

$$d_1(\gamma_1 w, \gamma_2 w') = \frac{1}{\|u\|} \sum_{i=1}^{N} |u(i) - u'(i)| \leq \frac{k}{\|u\|} \ .$$

[8] This definition of locality is not the same as the definition that appears in [6]. It is actually the same as the definition of the pairwise locality in [6]. The original definition of locality is of limited interest since it applies only to unnormalized algorithms.

[9] This stronger definition of rank locality is used for the characterization of the pSALSA algorithm (Theorem 9).

We note that $\|u\|$ is equal to the sum of the links in the graph. In the definition of stability we assumed that the number of links in the graph is $\omega(1)$. Therefore, $d_1(\gamma_1 w, \gamma_2 w') = o(1)$, which proves that L_1-pSALSA, and consequently pSALSA is stable. □

Now, let G be a graph that is "authority connected"; that is, the authority graph G_a, as defined in section 2.3, is connected. Then SALSA and pSALSA are equivalent on G. Let $\mathcal{G}_N^{AC}$ denote the family of authority connected graphs of size N. We have the following corollary.

Corollary 2 *The SALSA algorithm is stable on* $\mathcal{G}_N^{AC}$.

We originally thought that the Bayesian and Simplified Bayesian algorithms were also local. However, it turns out that they are neither local, nor strictly rank local. Indeed, it is true that *conditional* on the values of h_i, e_i, and a_j, the conditional distribution of a_k for $k \neq j$ is unchanged upon removing a link from i to j. However, the *unconditional* marginal distribution of a_k, and hence also its posterior mean $\widehat{a}_k$ (or even ratios $\widehat{a}_k/\widehat{a}_q$ for $q \neq j$), may still be changed upon removing a link from i to j. (Indeed, we have computed experimentally that $\widehat{a}_3/\widehat{a}_4$ may change upon removing a link from 1 to 2, even for a simple example with just four nodes.) Theorem 9 (proven below) implies that neither the Bayesian, nor the Simplified Bayesian Algorithm are strictly rank local, since (as shown experimentally) they not rank-matching with the pSALSA algorithm.

We now use locality and "label-independence" to prove a uniqueness property of the pSALSA algorithm.

Definition 12 *An algorithm is label-independent if permuting the labels of the graph nodes only causes the authority weights to be correspondingly permuted.*

All of our algorithms are clearly label-independent. Label-independence is a reasonable property, but one can define reasonable algorithms that are not label-independent. For example, consider the algorithm defined by Bharat and Henzinger [5]: when computing the authority weight of a node i, the hub weights of the nodes that belong to the same domain are averaged over the number of the nodes from that domain that point to node i. This algorithm is not label-independent.

Theorem 9 *Consider an algorithm A that is strictly rank local, monotone, and label-independent. Then A (and hence any normalized variant of A) and pSALSA are rank matching on $\overline{\mathcal{G}}_N$, for any $N \geq 3$.*

Proof: Let G be a graph of size $N \geq 3$, and let $a = A(G)$ be the weight function of algorithm A on graph G, and s be the weight function of pSALSA. We will be modifying G to form graphs G_1, and G_2 and we let a_1, and a_2 denote (respectively) the weight function of A on these graphs.

Let i and j be two nodes in G. If $s(i) = s(j)$, or equivalently (by definition of pSALSA) nodes i and j have the same number of in-links, then $\mathbf{I}_{as}(i,j) = 0$; therefore, nodes i and j do not violate the rank matching property. Without loss of generality assume that $s(i) > s(j)$, or equivalently that node i has more in-links than j. The set of nodes that point to i or j is decomposed as follows: there is a set of nodes C that point to both i and j; there is a set of nodes L that point only to node j; there is a set of cardinality R equal to L of nodes that point only to i; there is a non-empty set of nodes E that point to node i. Note that except for the set E, the other sets may be empty.

Let $k \neq i, j$ be an arbitrary node in the graph. We now perform the following change in graph G: remove all links that do not point to i or j, and make the nodes in L and C to point to node k. Let G_1 denote the resulting graph. Since A is strictly rank local, and the nodes i and j are not affected by the change, the order of nodes i and j is preserved. Furthermore, from the monotonicity of algorithm A, we have that $a_1(i) \geq a_1(k)$.

We will now prove that $a_1(k) = a_1(j)$. Assume that $a_1(k) < a_1(j)$. Let G_2 denote the graph that we obtain by removing all the links from set L to node i, and adding links from set R to node i. The graphs G_1 and G_2 are the same up to a label permutation that swaps the labels of nodes j and k. Therefore, $a_2(j) < a_2(k)$, which contradicts the assumption of strict rank locality. We reach the same contradiction if we assume that $a_1(k) > a_1(j)$. Thus, $a_1(k) = a_1(j)$, and $a_1(i) \geq a_1(j)$. Since A is strictly rank local, $a_1(i) \geq a_1(j) \Rightarrow a(i) \geq a(j)$.

Therefore, for all i, j, $\mathbf{I}_{as}(i,j) = 0$, that is, $d_r(a, s) = 0$, as required. □

In effect then, the conditions of Theorem 9 characterize pSALSA. All three conditions are necessary for the proof of the theorem. Assume that we discard the label independence condition. Now, define algorithm A, that assigns to each link a weight that depends on the label of the node the link originates from. The algorithm sets the authority weight of each node to be the sum of the link weights that point to this node. This algorithm is clearly monotone and local, however if the link weights are chosen appropriately, it will not be rank matching with pSALSA. Assume now that we discard of the monotonicity condition. Define an algorithm A, that assigns weight 1 to each node with odd in-degree, and weight 0 to each node with even in-degree. This algorithm is local and label independent, but it is clearly not rank matching with pSALSA. Monotonicity and label independence are clearly not sufficient for proving the theorem; we have provided examples of algorithms that are monotone and label independent, but not rank matching with the pSALSA (e.g. the Kleinberg algorithm).

8.4 Symmetry

Definition 13 *A "hubs and authorities" algorithm A is "symmetric" if inverting all the links in a graph simply interchanges the hub and authority values produced by the algorithm.*

We have by inspection:

Theorem 10 *The pSALSA and SALSA algorithms, the Kleinberg algorithm, the BFS algorithm, and the Simplified Bayesian algorithm are all symmetric. However, the Hub-Averaging algorithm, the Threshold algorithms, and the Bayesian algorithm are NOT symmetric.*

9 Summary

We have considered a number of known and some new algorithms which use the hypertext link structure of World Wide Web pages to extract information about the relative ranking of these pages. In particular, we have introduced two algorithms based on Bayesian statistical approach as well as a number of algorithms which are modifications of Kleinberg's seminal hubs and authority algorithm. Based on 8 different queries (5 presented here), we discuss some observed properties of each algorithm as well as relationships between the algorithms. We found (experimentally) that certain algorithms appear to be more "balanced", while others more "focused". The latter tend to be sensitive to the existence of tightly interconnected clusters, which may cause them to drift. The intersections between the lists of the top-ten results of the algorithms suggest that certain algorithms exhibit similar behavior and properties.

Motivated by the experimental observations, we introduced a theoretical framework for the study and comparison of link-analysis ranking algorithms. We formally defined (and gave some preliminary results for) the concepts of *monotonicity*, *stability* and *locality*, as well as various concepts of *distance* and *similarity* between ranking algorithms.

Our work leaves open a number of interesting questions. For example, are the Bayesian algorithms stable in any sense? What natural algorithms are similar (or rank similar) to each other? The two Bayesian algorithms open the door to the use of other statistical and machine learning techniques for ranking of hyper-linked documents. Furthermore, the framework we defined suggests a number of interesting directions for the theoretical study of ranking algorithms, which we have just begun to explore in this work. For example, in this work we proved that strict rank locality (together with monotonicity and label independence) implies rank matching with pSALSA. Such a result can be viewed as an axiomatic characterization of pSALSA and it would be interesting to know if other algorithms can be axiomatically characterized. In our work all the examples for instability are on disconnected graphs. It would be interesting to examine if instability can be proven for the class of connected graphs. Recent work has shown that stability is tightly connected with the spectral properties of the underlying graph [17, 18, 3, 1]. This seems a promising direction for proving stability results.

10 Acknowledgments

We would like to thank Ronald Fagin, Ronny Lempel, Alberto Mendelzon, and Shlomo Moran for valuable comments and corrections.

References

[1] D. Achlioptas, A. Fiat, A. Karlin, and F. McSherry. Web search through hub synthesis. In *Proceedings of the 42nd Foundation of Computer Science (FOCS 2001)*, Las Vegas, Nevada, 2001.

[2] AltaVista Company. AltaVista search engine. `http://www.altavista.com`.

[3] Y. Azar, A. Fiat, A. Karlin, F. McSherry, and J. Saia. Spectral analysis of data. In *Proceedings of the 33rd Symposium on Theory of Computing (STOC 2001)*, Hersonissos, Crete, Greece, 2001.

[4] J.M. Bernardo and A.F.M. Smith. *Bayesian Theory.* John Wiley & Sons, Chichester, England, 1994.

[5] K. Bharat and M. R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Research and Development in Information Retrieval*, pages 104–111, 1998.

[6] A. Borodin, G. O. Roberts, J. S. Rosenthal, and P. Tsaparas. Finding authorities and hubs from link structures on the World Wide Web. In *Proceedings of the 10th International World Wide Web Conference*, Hong Kong, 2001.

[7] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. In *Proceedings of the 7th International World Wide Web Conference*, Brisbane, Australia, 1998.

[8] D. Cohn and H. Chang. Learning to probabilistically identify authoritative documents. In *Proceedings of the 17th International Conference on Machine Learning*, pages 167–174, Stanford University, 2000.

[9] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

[10] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the Web. In *Proceedings of the 10th International World Wide Web Conference*, Hong Kong, 2001.

[11] W.R. Gilks, S. Richardson, and D.J. Spiegelhalter. *Markov Chain Monte Carlo in practice.* Chapman and Hall, London, 1996.

[12] Google. Google search engine. `http://www.google.com`.

[13] R. Kannan, S. Vempala, and A. Vetta. On clusterings: good, bad and spectral. In *Proceedings of the 41st Foundation of Computer Science (FOCS 2000)*, Redondo Beach, 2000.

[14] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of ACM (JASM)*, 46, 1999.

[15] R. Lempel and S. Moran. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. In *Proceedings of the 9th International World Wide Web Conference*, May 2000.

[16] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, Cambridge, England, June 1995.

[17] A. Y. Ng, A. X. Zheng, and M. I. Jordan. Link analysis, eigenvectors, and stability. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Seattle, Washington, USA, 2001.

[18] A. Y. Ng, A. X. Zheng, and M. I. Jordan. Stable algorithms for link analysis. In *Proceedings of the 24th International Conference on Research and Development in Information Retrieval (SIGIR 2001)*, New York, 2001.

[19] D. Rafiei and A. Mendelzon. What is this page known for? Computing web page reputations. In *Proceedings of the 9th International World Wide Web Conference*, Amsterdam, Netherlands, 2000.

[20] G.O. Roberts and J.S. Rosenthal. Markov chain Monte Carlo: Some practical implications of theoretical results (with discussion). *Canadian Journal of Statistics*, 26:5–31, 1998.

[21] G.O. Roberts and J.S. Rosenthal. Convergence of slice sampler Markov chains. *Journal of the Royal Statistical Society, Series B*, 61:643–660, 1999.

[22] A.F.M. Smith and G.O. Roberts. Bayesian computation via the Gibbs sampler and related Markov chain Monte Carlo methods (with discussion). *Journal of the Royal Statistical Society, Series B*, 55:3–24, 1993.

[23] L. Tierney. Markov chains for exploring posterior distributions (with discussion). *Annals of Statistics*, 22:1701–1762, 1994.

A Experiments

A.1 Query: abortion (Base Set size = 2293)

	Kleinberg	pSALSA	HubAvg	AThresh	HThresh	FThresh	BFS	SBayesian	Bayesian
1.	**P-1165**	P-717	**P-1165**	**P-1165**	P-717	P-1461	P-717	P-717	P-717
2.	**P-1184**	P-1461	**P-1184**	**P-1184**	P-1769	P-719	P-719	P-1461	**P-1192**
3.	**P-1193**	P-719	**P-1193**	**P-1193**	P-1461	P-1	P-1769	**P-1191**	**P-1165**
4.	**P-1187**	**P-1165**	**P-1187**	**P-1187**	P-718	P-717	P-1461	P-719	**P-1191**
5.	**P-1188**	**P-1184**	**P-1188**	**P-1188**	P-719	P-0	P-962	**P-1184**	**P-1193**
6.	**P-1189**	**P-1193**	**P-1189**	**P-1189**	P-1	P-115	P-0	**P-1165**	**P-1184**
7.	**P-1190**	**P-1187**	**P-1190**	**P-1190**	P-0	P-607	P-2	**P-1192**	**P-1189**
8.	**P-1191**	**P-1188**	**P-1191**	**P-1191**	P-2515	P-1462	P-718	**P-1193**	**P-1188**
9.	**P-1192**	**P-1189**	**P-1192**	**P-1192**	P-115	P-2	P-1325	**P-1188**	**P-1187**
10.	P-1948	**P-1190**	P-1948	P-1948	P-962	P-1567	P-1522	**P-1187**	**P-1190**

Index	pop	URL	Title
P-0	3	www.gynpages.com	Abortion Clinics OnLine
P-1	2	www.prochoice.org	NAF - The Voice of Abortion Providers
P-2	2	www.cais.com/agm/main	The Abortion Rights Activist Home Page
P-115	2	www.ms4c.org	Medical Students for Choice
P-607	1	www.feministcampus.org	Feminist Campus Activism Online: Welcome
P-717	6	www.nrlc.org	National Right to Life Organization
P-718	2	www.hli.org	Human Life International (HLI)
P-719	5	www.naral.org	NARAL: Abortion and Reproductive Rights: ...
P-962	2	www.prolife.org/ultimate	Empty title field
P-1165	6	www5.dimeclicks.com	DimeClicks.com - Web and Marketing Solutions
P-1184	6	www.amazon.com/...../youdebatecom	Amazon.com–Earth's Biggest Selection
P-1187	6	www.amazon.com/...../top-sellers.html	Amazon.com–Earth's Biggest Selection
P-1188	6	www.amazon.com/.../software/home.html	Amazon.com Software
P-1189	5	www.amazon.com/.../hot-100-music.html	Amazon.com–Earth's Biggest Selection
P-1190	5	www.amazon.com/.../gifts.html	Amazon.com–Earth's Biggest Selection
P-1191	5	www.amazon.com/.....top-100-dvd.html	Amazon.com–Earth's Biggest Selection
P-1192	5	www.amazon.com/...top-100-video.html	Amazon.com–Earth's Biggest Selection
P-1193	6	rd1.hitbox.com/.......	HitBox.com - hitbox web site ....
P-1325	1	www.serve.com/fem4life	Feminists For Life of America
P-1461	5	www.plannedparenthood.org	Planned Parenthood Federation of America
P-1462	1	http://www.rcrc.org	The Religious Coalition for Reproductive Choice
P-1522	1	www.naralny.org	NARAL/NY
P-1567	1	http://www.agi-usa.org	The Alan Guttmacher Institute: Home Page
P-1769	2	www.priestsforlife.org	Priests for Life Index
P-1948	3	www.politics1.com/issues.htm	Politics1: Hot Political Debates & Issues
P-2515	1	www.ohiolife.org	Ohio Right To Life

	Kleinberg	SALSA	HubAvg	AThresh	HThresh	FThresh	BFS	SBayesian	Bayesian
Kleinberg	10	7	10	10	0	0	0	7	9
SALSA	7	10	7	7	3	3	3	8	8
HubAvg	10	7	10	10	0	0	0	7	9
AThresh	10	7	10	10	0	0	0	7	9
HThresh	0	3	0	0	10	6	7	3	1
FThresh	0	3	0	0	6	10	5	3	1
BFS	0	3	0	0	7	5	10	3	1
SBayesian	7	8	7	7	3	3	3	10	8
Bayesian	9	8	9	9	1	1	1	8	10

A.2 Query: +net +censorship (Base Set size = 2947)

	Kleinberg	pSALSA	HubAvg	AThresh	HThresh	FThresh	BFS	SBayesian	Bayesian
1.	**P-375**	**P-371**	**P-371**	**P-371**	**P-375**	**P-375**	**P-371**	**P-371**	**P-375**
2.	**P-3163**	P-1440	**P-2874**	**P-375**	P-1344	P-1344	**P-375**	P-1440	**P-371**
3.	**P-3180**	**P-375**	**P-2871**	**P-2871**	**P-3130**	**P-3130**	P-1299	**P-375**	**P-3180**
4.	**P-3177**	**P-2874**	**P-2873**	**P-2874**	**P-3131**	**P-3131**	P-1440	**P-2874**	**P-3177**
5.	**P-3173**	**P-2871**	**P-2659**	**P-3536**	**P-3132**	**P-3132**	**P-2871**	P-1299	**P-3163**
6.	**P-3172**	P-1299	**P-375**	**P-2873**	**P-3133**	**P-3133**	**P-2874**	**P-2871**	**P-3173**
7.	**P-3132**	**P-3536**	**P-3536**	**P-2659**	**P-3135**	**P-3135**	**P-3536**	**P-3536**	**P-3166**
8.	P-3193	P-1712	**P-2639**	**P-2639**	**P-3161**	**P-3161**	**P-1802**	P-1712	P-3193
9.	**P-3170**	P-268	P-1440	**P-2639**	**P-3162**	**P-3162**	**P-2639**	P-268	**P-3168**
10.	**P-3166**	P-1445	**P-2867**	P-1440	**P-3163**	**P-3163**	P-452	P-1445	**P-3132**

Index	pop	URL	Title
P-268	2	www.epic.org	Electronic Privacy Information Center
P-371	6	www.yahoo.com	Yahoo!
P-375	9	www.cnn.com	CNN.com
P-452	1	www.mediachannel.org	MediaChannel.org – A Global Network ....
P-1299	3	www.eff.org/blueribbon.html	EFF Blue Ribbon Campaign
P-1344	2	www.igc.apc.org/peacenet	PeaceNet Home
P-1440	5	www.eff.org	EFF ... - the Electronic Frontier Foundation
P-1445	2	www.cdt.org	The Center for Democracy and Technology
P-1712	2	www.aclu.org	ACLU: American Civil Liberties Union
P-1802	1	ukonlineshop.about.com	Online Shopping: UK
P-2639	3	www.imdb.com	The Internet Movie Database (IMDb).
P-2659	2	www.altavista.com	AltaVista - Welcome
P-2867	1	home.netscape.com	Empty title field
P-2871	5	www.excite.com	My Excite Start Page
P-2873	2	www.mckinley.com	Welcome to Magellan!
P-2874	5	www.lycos.com	Lycos
P-3130	2	www.city.net/countries/kyrgyzstan	Excite Travel
P-3131	2	www.bishkek.su/krg/Contry.html	ElCat. 404: Not Found.
P-3132	4	www.pitt.edu/~cjp/rees.html	REESWeb: Programs:
P-3133	2	www.ripn.net	RIPN
P-3135	2	www.yahoo.com/.../Kyrgyzstan	Yahoo! Regional Countries Kyrgyzstan
P-3161	2	151.121.3.140/fas/fas-publications/...	Error 404 Redirector
P-3162	2	www.rferl.org/BD/KY	RFE/RL Kyrgyz Service : News
P-3163	4	www.usa.ft.com	Empty title field
P-3166	2	www.pathfinder.com/time/daily	TIME.COM
P-3168	1	www.yahoo.com/News	Yahoo! News and Media
P-3170	1	www.financenet.gov	...FinanceNet is the government's official home...
P-3172	1	www.oecd.org	OECD Online
P-3173	2	www.worldbank.org	The World Bank Group
P-3177	2	www.envirolink.org	EnviroLink Network
P-3180	3	www.lib.utexas.edu/.../Map_collection	PCL Map Collection
P-3193	2	www.wiesenthal.com	Simon Wiesenthal Center
P-3536	5	www.shareware.com	CNET.com - Shareware.com

	Kleinberg	SALSA	HubAvg	AThresh	HThresh	FThresh	BFS	SBayesian	Bayesian
Kleinberg	10	1	1	2	3	3	1	1	8
SALSA	1	10	6	6	1	1	7	10	2
HubAvg	1	6	10	9	1	1	7	6	2
AThresh	2	6	9	10	1	1	7	6	3
HThresh	3	1	1	1	10	10	1	1	3
FThresh	3	1	1	1	10	10	1	1	3
BFS	1	7	7	7	1	1	10	7	2
SBayesian	1	10	6	6	1	1	7	10	2
Bayesian	8	2	2	3	3	3	2	2	10

A.3 Query: `Movies (Base Set size = 5757)`

	Kleinberg	pSALSA	HubAvg	AThresh	HThresh	FThresh	BFS	SBayesian	Bayesian
1.	**P-678**	P-999	P-999	P-999	**P-678**	**P-1989**	P-999	P-999	P-999
2.	**P-2268**	P-2832	P-2832	P-2832	P-2266	P-2832	**P-1911**	P-2832	P-2832
3.	**P-2304**	P-6359	**P-2101**	P-6359	**P-2263**	**P-1980**	P-2827	P-6359	P-2827
4.	**P-2305**	P-2827	**P-803**	P-2827	**P-2264**	**P-1983**	P-2803	P-2827	P-6359
5.	**P-2306**	**P-2120**	**P-1539**	P-2838	**P-2265**	**P-1984**	P-5470	**P-1374**	**P-678**
6.	**P-2308**	**P-1374**	**P-1178**	P-6446	**P-2268**	**P-1986**	**P-2120**	**P-2120**	P-2838
7.	**P-2310**	**P-803**	P-6359	P-5	**P-2280**	**P-1987**	**P-4577**	**P-803**	P-2266
8.	P-2266	**P-1539**	P-1082	P-2803	**P-2299**	**P-1993**	P-5	P-6446	**P-2268**
9.	**P-2325**	P-6446	P-2827	P-2839	**P-2300**	**P-1995**	P-2838	**P-1539**	**P-2308**
10.	**P-2299**	P-2838	P-6446	P-2840	**P-2301**	**P-3905**	P-4534	P-2838	**P-2330**

Index	pop	Title	URL
P-5	2	www.movies.com	Movies.com
P-678	3	chatting.about.com	Empty title field
P-803	3	www.google.com	Google
P-999	6	www.moviedatabase.com	The Internet Movie Database (IMDb).
P-1082	1	www.amazon.com/	Amazon.com–Earth's Biggest Selection
P-1178	1	www.booksfordummies.com	Empty title field
P-1374	2	www.onwisconsin.com	On Wisconsin
P-1539	3	206.132.25.51	Washingtonpost.com - News Front
P-1911	1	people2people.com/...nytoday	People2People.com - Search
P-1980	1	newyork.urbanbaby.com/nytoday	Kids & Family
P-1983	1	tunerc1.va.everstream.com/nytoday/	Empty title field
P-1984	1	nytoday.opentable.com/	OpenTable
P-1986	1	www.nytimes.com/.../jobmarket	The New York Times: Job Market
P-1987	1	www.cars.com/nytimes	New York Today cars.com - new and used car ...
P-1989	1	www.nytodayshopping.com	New York Today Shopping - Shop for computers, ...
P-1993	1	www.nytimes.com/.../nytodaymediakit	New York Today - Online Media Kit
P-1995	1	http://www.nytimes.com/subscribe...	The New York Times on the Web
P-2101	1	www2.ebay.com/aw/announce.shtml	eBay Announcement Board
P-2120	3	www.mylifesaver.com	welcome to mylifesaver.com
P-2263	1	clicks.about.com/...nationalinterbank	Banking Center
P-2264	1	clicks.about.com/	Credit Report, Free Trial Offer
P-2265	1	membership.about.com/...	Member Center
P-2266	3	home.about.com/movies	About - Movies
P-2268	3	a-zlist.about.com	About.com A-Z
P-2280	1	sprinks.about.com	Sprinks : About Sprinks
P-2299	2	home.about.com/aboutaus	About Australia
P-2300	1	http://home.about.com/aboutcanada	About Canada
P-2301	1	http://home.about.com/aboutindia	About India
P-2304	1	home.about.com/arts	About - Arts/Humanities
P-2305	1	home.about.com/autos	About - Autos
P-2306	1	home.about.com/citiestowns	About - Cities/Towns
P-2308	2	home.about.com/compute	About - Computing/Technology
P-2310	1	home.about.com/education	About - Education
P-2325	1	home.about.com/musicperform	About - Music/Performance
P-2330	1	home.about.com/recreation	About - Recreation/Outdoors
P-2803	2	www.allmovie.com	All Movie Guide
P-2827	6	www.film.com	Film.com Movie Reviews, News, Trailers...
P-2832	6	www.hollywood.com	Hollywood.com - Your entertainment source...
P-2838	5	www.mca.com	Universal Studios
P-2839	1	www.mgmua.com	MGM - Home Page
P-2840	1	www.miramax.com	Welcome to the Miramax Cafe
P-3905	1	http://theadvocate.webfriends.com	Empty title field
P-4534	1	www.aint-it-cool-news.com	Ain't It Cool News
P-4577	1	go.com	GO.com
P-5470	1	www.doubleaction.net	Double Action - Stand. Point. Laugh.
P-6359	5	www.paramount.com	Paramount Pictures - Home Page
P-6446	4	www.disney.com	Disney.com – Where the Magic Lives Online!

	Kleinberg	SALSA	HubAvg	AThresh	HThresh	FThresh	BFS	SBayesian	Bayesian
Kleinberg	10	0	0	0	4	0	0	0	4
SALSA	0	10	7	6	0	0	5	10	5
HubAvg	0	7	10	5	0	0	3	7	4
AThresh	0	6	5	10	0	0	6	6	5
HThresh	4	0	0	0	10	0	0	0	3
FThresh	0	0	0	0	0	10	0	0	0
BFS	0	5	3	6	0	0	10	5	4
SBayesian	0	10	7	6	0	0	5	10	5
Bayesian	4	5	4	5	3	0	4	5	10

A.4 Query: `genetic (Base Set size = 3468)`

	Kleinberg	pSALSA	HubAvg	AThresh	HThresh	FThresh	BFS	SBayesian	Bayesian
1.	P-2187	P-2187	P-2187	P-2187	P-2187	P-2187	P-2187	P-2187	P-2187
2.	P-1057	P-258	P-1057	P-1057	P-1057	P-1057	P-3932	P-258	P-1057
3.	P-2168	P-1057	P-3932	P-2168	P-2168	P-2168	**P-1538**	P-1057	P-2168
4.	P-2200	P-3932	P-2095	P-2200	P-2200	P-2200	P-1057	P-3932	P-2095
5.	P-2219	P-2095	P-2168	P-2219	P-2219	P-2219	P-2095	P-2095	P-2200
6.	P-2199	**P-1538**	P-2186	P-2095	P-2199	P-2095	P-258	**P-1538**	P-2219
7.	P-2095	P-2	P-941	P-3932	P-2186	P-3932	P-2168	P-2	P-3932
8.	P-2186	P-2168	P-0	P-2199	P-2095	P-2199	P-2200	P-2168	P-2199
9.	P-2193	P-941	P-2200	P-2186	P-2193	P-2186	P-2	P-941	P-2186
10.	P-3932	P-23	P-2199	P-2193	P-3932	P-2193	P-2199	P-2200	P-2193

Index	pop	URL	Title
P-0	1	www.geneticalliance.org	Genetic Alliance, Washington, DC
P-2	3	www.genetic-programming.org	genetic-programming.org-Home-Page
P-23	1	www.geneticprogramming.com	The Genetic Programming Notebook
P-258	3	www.aic.nrl.navy.mil/galist	The Genetic Algorithms Archive
P-941	3	www3.ncbi.nlm.nih.gov/Omim	OMIM Home Page – Online Mendelian Inheritance in Man
P-1057	9	gdbwww.gdb.org	The Genome Database
P-1538	3	www.yahoo.com	Yahoo!
P-2095	9	www.nhgri.nih.gov	National Human Genome Research Institute (NHGRI)
P-2168	9	www-genome.wi.mit.edu	Welcome To the ..... Center for Genome Research
P-2186	6	www.ebi.ac.uk	EBI, the European Bioinformatics Institute ....
P-2187	9	www.ncbi.nlm.nih.gov	NCBI HomePage
P-2193	5	www.genome.ad.jp	GenomeNet WWW server
P-2199	7	www.hgmp.mrc.ac.uk	UK MRC HGMP-RC
P-2200	8	www.tigr.org	The Institute for Genomic Research
P-2219	5	www.sanger.ac.uk	The Sanger Centre Web Server
P-3932	9	www.nih.gov	National Institutes of Health (NIH)

	Kleinberg	pSALSA	HubAvg	AThresh	HThresh	FThresh	BFS	SBayesian	Bayesian
Kleinberg	10	5	8	10	10	10	7	6	10
pSALSA	5	10	6	5	5	5	8	9	5
HubAvg	8	6	10	8	8	8	7	7	8
AThresh	10	5	8	10	10	10	7	6	10
HThresh	10	5	8	10	10	10	7	6	10
FThresh	10	5	8	10	10	10	7	6	10
BFS	7	8	7	7	7	7	10	9	7
SBayesian	6	9	7	6	6	6	9	10	6
Bayesian	10	5	8	10	10	10	7	6	10

A.5 Query: `+computational +geometry (Base Set size = 1226)`

	Kleinberg	pSALSA	HubAvg	AThresh	HThresh	FThresh	BFS	SBayesian	Bayesian
1.	P-161	P-161	**P-634**	P-161	P-0	P-161	P-0	P-161	P-161
2.	P-0	P-1	**P-632**	P-0	P-161	P-0	P-161	P-1	P-0
3.	P-1	P-0	**P-633**	P-1	P-1	P-1	P-1	P-0	P-1
4.	P-162	P-3	P-1406	P-162	P-162	P-162	P-300	P-3	P-162
5.	P-3	P-280	P-161	P-3	P-280	P-280	P-299	P-280	P-3
6.	P-280	**P-634**	P-162	P-280	P-3	P-3	P-162	**P-634**	P-280
7.	P-275	P-162	P-1369	P-275	P-275	P-275	P-3	P-162	P-275
8.	P-299	P-2	P-351	P-299	P-299	P-299	P-280	P-2	P-299
9.	P-300	**P-632**	**P-1308**	P-300	P-300	P-848	P-375	**P-633**	P-300
10.	P-848	**P-633**	P-1	P-848	P-848	P-300	P-551	**P-632**	P-848

Index	pop	URL	Title
P-0	8	www.geom.umn.edu/software/cglist	Directory of Computational Geometry Software
P-1	9	www.cs.uu.nl/CGAL	The former CGAL home page
P-2	2	link.springer.de/link/service/journals/00454	LINK: Peak-time overload
P-3	8	www.scs.carleton.ca/~csgs/resources/cg.html	Computational Geometry Resources
P-161	9	www.ics.uci.edu/~eppstein/geom.html	Geometry in Action
P-162	9	www.ics.uci.edu/~eppstein/junkyard	The Geometry Junkyard
P-275	5	www.ics.uci.edu/~eppstein	David Eppstein
P-280	8	www.geom.umn.edu	The Geometry Center Welcome Page
P-299	6	www.mpi-sb.mpg.de/LEDA/leda.html	LEDA - Main Page of LEDA Research
P-300	6	www.cs.sunysb.edu/~algorith	The Stony Brook Algorithm Repository
P-351	1	http://www.ics.uci.edu/~eppstein/gina/...	Geometry Publications by Author
P-375	1	graphics.lcs.mit.edu/~seth	Seth Teller
P-551	1	www.cs.sunysb.edu/~skiena	Steven Skiena
P-632	3	www.w3.org/Style/CSS/Buttons	CSS button
P-633	3	jigsaw.w3.org/css-validator	W3C CSS Validation Service
P-634	3	validator.w3.org	W3C HTML Validation Service
P-848	5	www.inria.fr/prisme/....../cgt	CG Tribune
P-1308	1	http://netlib.bell-labs.com/...compgeom	/netlib/compgeom
P-1369	1	http://www.adobe.com/...	Adobe Acrobat Reader
P-1406	1	www.informatik.rwth-aachen.de/.....	Department of Computer Science, Aachen

	Kleinberg	SALSA	HubAvg	AThresh	HThresh	FThresh	BFS	SBayesian	Bayesian
Kleinberg	10	6	3	10	10	10	8	6	10
SALSA	6	10	6	6	6	6	6	10	6
HubAvg	3	6	10	3	3	3	3	6	3
AThresh	10	6	3	10	10	10	8	6	10
HThresh	10	6	3	10	10	10	8	6	10
FThresh	10	6	3	10	10	10	8	6	10
BFS	8	6	3	8	8	8	10	6	8
SBayesian	6	10	6	6	6	6	6	10	6
Bayesian	10	6	3	10	10	10	8	6	10