A Secure Online Algorithm for Link Analysis on Weighted Graph

Yitao Duan, Jingtao Wang, Matthew Kam and John Canny

Computer Science Division
University of California, Berkeley, CA 94720
{duan, jingtaow, mattkm, jfc}@EECS.Berkeley.EDU

Abstract

Link analysis algorithms have been used successfully on hyperlinked data to identify authoritative documents and retrieve other information. However, existing link analysis algorithms such as HITS suffer two major limitations: (1) they only work in environments with explicit hyperlinked structure such as www or social network and (2) they fail to capture the rich information that is encoded by patterns of user access. In this paper we propose the use of weighted graph that is generated and updated via analysis of patterns of user access to address both issues. We then present a generalized HITS algorithm that is suitable for such an approach. The algorithm uses the idea of “lazy update” to amortize cost across a number of updates while still providing accurate ranking to users in real-time. We proved the convergence of the new online algorithm and evaluated its benefit using simulation. Finally we devised a scheme that makes the algorithm distributed and privacy preserving using cryptographic techniques thus making it really acceptable in settings such as collaborative work and online community.

1 Introduction

Link analysis algorithms have been used successfully on hyperlinked data to identify authoritative documents and retrieve other information. For instance, the expertise location problem [1, 3, 4, 5] is to find a person in a community who is knowledgeable (and authoritative) in an area. Several approaches [1, 3, 4] construct an explicit social network between individuals, based on email or similar logs, and then use graphical analysis to locate the relevant experts. Similarly, the document ranking problem is to determine the relative levels of “authoritativeness” among a collection of documents. Link analysis algorithms have been used in such environments to uncover such information [6, 7].

The primary drawback to the above approaches is the need for explicit structure about the social relationships between individuals and the hyperlinks among documents, which do not necessarily exist. For instance, in a computer-mediated environment, a group of individuals could be using tools like software applications to access documents collaboratively, and there is neither an explicit social network representing how each individual is related to others, nor hyperlinks among documents. However, in such context, there are still compelling needs in identifying domain experts and authoritative documents.

Another inadequacy of such algorithms, as Kleinberg acknowledged [6], is that they only make use of the structural information about the graph as defined by the links, and fail to capture patterns of user access which encode essential information about the user’s attitude toward the document. The intuition behind the link analysis algorithms is that the link structure encodes important information about the nodes. For example, according to [6], the links among documents, be it hyperlinks on www or citations among academic papers, are constructed consciously by the authors of the documents and represent the authors’ “endorsement” toward the authority of documents pointed to by the links and the HITS algorithm [6] can uncover such information to produce a ranking of documents according their level of
authority. We believe that a similar principle also holds with patterns of user access: the way a user accesses a document could reflect his/her opinion about it. Meanwhile, a user’s level of expertise can also be reflected by the documents that he/she accesses. There is a mutually reinforcing relationship between these two measures, which maps naturally to what Kleinberg denotes “hub” and “authority” [6]: a person is more likely to be an expert in an area if he/she reads more authoritative documents and a document is more likely to be authoritative if it is read by many experts.

In this paper we propose an approach to address both limitations simultaneously and describe an algorithm that is suitable for such purpose. Notice that access pattern and link structure are not mutually exclusive types of information. Rather, access pattern can complement or even define the other. Our approach uses weighted graphs to model the relationship among users and/or documents where weights encode user access. In situations where no explicit link structure exists, access pattern effectively defines the structure and a weighted graph can be constructed for use with link analysis algorithms. Where there is explicit link structure, weights obtained from access analysis can be used to augment existing graph and uncover more information.

Using weights in identifying authoritative documents is not new [9]. The novelty of this paper is that we propose the use of weights to model user access and construct the link structure. This enables us to apply link analysis algorithms in settings where no such structure exists. However, including patterns of access into consideration has two implications: (1) instead of a static system, the graph becomes dynamic. Users’ access patterns are temporally accumulating (new records won’t overwrite old ones; instead, they are appended). The model changes as more data is observed; and (2) user privacy becomes an issue due to the sensitive nature of the user’s information used to construct the graph. (1) may also mandate that the system services users’ query in real-time as there is no end to the accumulation of observations on access pattern.

To address these new issues, we devised Secure OnlineHITS, a distributed version and enhancement of Kleinberg HITS algorithm that (1) amortizes cost across a number of updates by using “lazy updates”, which makes it more suitable for dynamic environments; and (2) uses cryptographic techniques to preserve user’s privacy while performing the computation. To make it concrete, we describe the algorithm in the context of document and expertise ranking. However, it is general enough to be applied to other situations where link analysis is appropriate. We use the term document in a broad sense. It refers to any information that can be identified, accessed and analyzed as a unit. For example, a web page or an image can all be classified as a document.

In the rest of the paper, we first review the original HITS algorithm in Section 2. We then discuss the construction of a graph by modeling of user access using a weight function in Section 2. In Section 4 we derive an online version of the HITS algorithm to make it more efficient to run in a dynamic environment on accumulated data. Evaluations are presented in Section 5. Finally in Section 6 we discuss privacy and security issues in running such kind of user activity analysis and describe our privacy-enhanced algorithm based on public-key encryption.

2 A Review of HITS

Kleinberg’s HITS algorithm [6] is a well-known link analysis algorithm that identifies “authoritative” or “influential” webpages in a hyperlinked environment. Intuitively, by thinking of a hyperlink as a citation, a webpage $i$ is more of an authority (i.e. highly-referenced page) as compared to webpage $j$ if there are more hyperlinks entering $i$ from hub webpages, where a hub is simply a webpage that is a valuable source of links to other webpages. Likewise, a webpage $i$ is a better hub than webpage $j$ if there are more hyperlinks exiting $i$ into authoritative webpages. Given a set of $n$ webpages, HITS first constructs the corresponding $n$-by-$n$ adjacency matrix $A$, such that the element in row $i$ and column $j$ of $A$ is 1 if there exists a hyperlink from webpage $i$ to webpage $j$, 0 otherwise. HITS then iterates the
following equations:
\[ x^{(t+1)} = A^T y^{(t)} = (A^T A)x^{(t)} \]  
\[ y^{(t+1)} = Ax^{(t+1)} = (AA^T)y^{(t)} \]  

Where the \(i\)-th element of \(x\) denotes the authoritativeness of webpage \(i\) and the \(i\)-th element of \(y\) denotes the value of webpage \(i\) as a hub. With the vectors \(x\) and \(y\) initialized as vectors of ones and renormalized to unit length at every iteration, as \(t\) approaches infinity, \(x^{(t+1)}\) and \(y^{(t+1)}\) approach \(x^*\) and \(y^*\), the principal eigenvectors of \(A^T A\) and \(AA^T\), respectively.

Even though HITS is originally intended to locate hubs and authorities in a hyperlinked environment, we observe that hubs and authorities map very well to the users and documents in access based link analysis and the relationship of mutual reinforcement still holds as mentioned in Section 1.

3 From Access to Weighted Graph

Using access analysis enables us to construct a graph in environments where no such structure exists. We assume we can observe users’ document access using tools like client side logger. Such tools are available from a number of sources. In particular, we have developed a version of our own that runs on Windows platform and can log documents and webpages the user accesses. Of course such tools have serious privacy implications and we will address such issue in Section 6.

The system consciously logs the users’ activities as tuples of the form \(<i, j>\), which denotes the fact that user \(i\) accesses document \(j\). These log entries represent tacit data about the collaborative context because they do not directly encode the links between users nor documents. Given this activity log, we can construct a bipartite factor graph, such that vertices in one partition represent the users and vertices in the other partition represent the documents. An edge \((i, j)\) exists and has weight \(w_{i,j}\) iff user \(i\) has accessed document \(j\) as reflected in the activity log.

3.1 Convergence of Weighted HITS

Suppose we replace the 0-1 valued element \(A_{ij}\) in the adjacency matrix \(A\) with a weight function \(w(i, j)\). Here \(w(i, j)\) represents user \(i\)’s vote to document \(j\) based on his access pattern to documents. Let \(w(i, j)\) equals to zero if user \(i\) had never accessed document \(j\) and let \(w(i, j)\) be a positive value if document \(j\) had been browsed by user \(i\). First we introduce the following two lemmas from [8].

Lemma 1 If \(M\) is a symmetric matrix and \(v\) is a vector not orthogonal to the principal eigenvector \(\lambda_1\) of \(M\), then the unit vector in the direction of \(M^kV\) converges to \(\lambda_1\) as \(k\) increases without bound.

Lemma 2 If a symmetric \(M\) has only non-negative elements, the principal eigenvector of \(M\) has only non-negative entries.

According to the definition of \(w(i, j)\), it’s easy to see that matrix \(A\) has only non-negative values and the symmetric matrix \(A^T A\) and \(AA^T\) have only non-negative values, thus the principal eigenvectors of \(A^T A\) and \(AA^T\) have only non-negative entries (lemma 2).

If we use a non-negative values vector \(x\), since \(x\) is not orthogonal to the eigenvector of \(AA^T\) which has only non-negative entries, the sequence \(\{y_k\}\) converges to a limit \(y^*\) (lemma 1). Similarly we can prove that the sequence \(\{x_k\}\) converges to a limit \(x^*\).

4 Online HITS

Access based graph construction and link analysis introduces a number of issues of its own such as frequent update, distributed data sources, data security and user privacy concerns, etc. An algorithm
along cannot address all these issues. But a properly designed algorithm can make addressing them a lot easier. In this section we describe a link analysis algorithm that works incrementally as data is being added. We use the idea of “lazy update” to avoid updating and running of the expensive computation so that we can amortize the cost across a number of updates while still maintaining enough precision.

4.1 Basic Approach

As shown in Section 1 and 3, the intuition behind HITS fits very well to our application. However, the algorithm is too expensive to run on every update, which can be quite frequent. Recall that the rankings we are seeking, \( x \) and \( y \), correspond to the the principal eigenvectors of \( A^T A \) and \( A A^T \), respectively. A key observation is that a single update to the user access traffic corresponds to a perturbation to the \( A \) matrix. Depending on the weight function selected, it can perturbate a single element or a row of \( A \). In either case the perturbation is local. This perturbation will cause variation to the principal eigenvector of \( A^T A \) (and \( A A^T \)). If we can find the relationship between the variation of \( x \) and \( y \) and the perturbation to \( A \), we can check each update to see if it will cause too much variation to \( x \) and \( y \). If the change is within acceptable precision, we can postpone applying the update thus avoiding running HITS for it. When the accumulated updates cause too much perturbation, we apply them together and run HITS once. This is essentially an approximation to HITS that amortizes its cost across multiple updates. We denote such an algorithm Online HITS. Another advantage of this approach is that service of user queries and updating \( A \) and running of HITS can be made separate. The system can update \( A \) and run HITS in background and continue servicing user queries with old results that we are confident to be within certain range from the the latest ones. Users can enjoy nonblocking service.

Similar issues have been discussed in the context of stability of the HITS algorithm [12, 13]. However, there is a subtle but significant difference between our approach and theirs: we are not concerned with the incompleteness of our data or the stability of the results. For us, the everlasting accumulation of data is an inherent feature of our system and the results we produce are the “best guess” based on the data we have so far. It is perfectly alright for the results to undergo dramatic change, which reflects the update of the system’s knowledge about the world. Rather, we are interested in the bound of the change so that we can perform the tasks more efficiently. In addition, the conclusions in [12, 13] only apply to unweighted graphs represented by adjacency matrices. The theorem we describe below is applicable to any weighted graph.

Online HITS is based on the following theorem:

**Theorem 1** Let \( S = A^T A \) be a symmetric matrix. Let \( a^* \) be the principal eigenvector and \( \delta \) the eigengap\(^1 \) of \( S \). Let \( E_S \) be a symmetric perturbation to \( S \). We use \( \| \cdot \|_F \) to denote the Frobenius norm\(^2 \). For any \( \epsilon > 0 \), if \( \| E_S \|_F \leq \min \{ \epsilon \delta \frac{\sqrt{1 + \sqrt{2}}}{4 \sqrt{2}}, \frac{\delta}{2 \sqrt{2}} \} \), then the principal eigenvector \( \tilde{a}^* \) of the perturbed matrix \( \tilde{S} = S + E_S \) satisfies

\[
\| a^* - \tilde{a}^* \|_2 \leq \epsilon
\]

This theorem gives us a way to test the perturbation and bound the error on the principal eigenvector. The proof is similar to that presented in [13] and is given in appendix.

There are two subtle issues that need to be addressed before we can use this theorem to construct an online HITS algorithm, namely the computations of eigengap \( \delta \) and perturbation \( \| E_S \|_F \). They have to be performed efficiently otherwise the cost of computing them would offset the saving of not running HITS. They will be addressed in the following subsections.

---

1Eigengap is defined to be the difference between the largest and the second largest eigenvalues.

2The Frobenius norm of a matrix \( X \) is defined by \( \| X \|_F = (\sum_{i} \sum_{j} (X_{ij})^2)^{1/2} \)
4.2 Computation of Eigengap

A straightforward way of computing eigengap $\delta$ is to calculate $\lambda_1$ and $\lambda_2$, the largest and the second largest eigenvalues, and take the difference. The original HITS algorithm is essentially a power method to compute the principal eigenvector of $S$. It can be revised easily, without adding complexity, to produce $\lambda_1$ and $\lambda_2$ as byproducts. Two modifications to the original HITS algorithm are introduced:

1. Instead of finding only the principal eigenvector, find the two eigenvectors corresponding to $\lambda_1$ and $\lambda_2$. This can be done by using “block power method” ([14], pp. 289). Concretely, start with two orthogonal vectors, multiply them all by $S$, then apply Gram-Schmidt to orthogonalize them. This is a single step. Iterate until they converge.

2. HITS normalizes the vector at each step to unit length. This is not necessarily the only choice to ensure convergence. Instead, we normalize each vector by dividing them by their first non-zero element. They still converge to the two eigenvectors and the scaling factors converge to $\lambda_1$ and $\lambda_2$ ([14], pp. 289).

The above modifications introduce extra computation of one eigenvector and Gram-Schmidt orthogonalization. The former doubles the work of HITS and the latter is $O(n)$. The total complexity is the same as HITS: $O(mn)$.

4.3 Upper Bound of $\|E_S\|_F$

Let $E$ be perturbation to matrix $A$ (This is our update to the graph). Then $\tilde{A} = A + E$ and $\tilde{S} = (A + E)^T (A + E) = A^T A + A^T E + E^T A + E^T E$. Let $E_S = A^T E + E^T A + E^T E$. We know for Frobenius norm (actually for any norms) $\|X + Y\|_F \leq \|X\|_F + \|Y\|_F$. So $\|E_S\|_F \leq 2 \|A^T E\|_F + \|E^T E\|_F$. This bound involves matrix multiplication which we try to avoid. Note that the purpose of our online HITS is to postpone running the algorithm so that we can save some computation. This means that we will accumulate a number of updates (since the last time we update $A$ and run HITS). Even though each single update is local and involve only one element or one row of $A$, all the accumulated updates will affect a number of $A$’s elements. This means $E$ can be sparse but unlikely to have only single non-zero element or a row. Let $E(t)$ be the accumulated unapplied update matrix after we observed $t$th update (we reset the counting each time we apply updates). $E(t) = E(t - 1) + \Delta(t)$ where $\Delta(t)$ has only one non-zero element or row. We have

$$\|E_S(t)\|_F \leq 2\|A^T E(t)\|_F + \|E(t)^T E(t)\|_F \quad (3)$$

where

$$\|A^T E(t)\|_F = \|A^T (E(t - 1) + \Delta(t))\|_F \leq \|A^T E(t - 1)\|_F + \|A^T \Delta(t)\|_F \quad (4)$$

and

$$\|E(t)^T E(t)\|_F = \|(E(t - 1) + \Delta(t))^T (E(t - 1) + \Delta(t))\|_F$$

$$= \|E(t - 1)^T E(t - 1) + E(t - 1) E(t - 1)^T \Delta(t) + \Delta(t)^T E(t - 1) + \Delta(t)^T \Delta(t)\|_F$$

$$\leq \|E(t - 1)^T E(t - 1)\|_F + 2\|E(t - 1)^T \Delta(t)\|_F + \|\Delta(t)^T \Delta(t)\|_F \quad (5)$$

The three equations above give us a way to compute the upper bound on $\|E_S\|_F$ recursively. Namely we can keep running updates on the upper bounds of $\|A^T E(t - 1)\|_F$ and $\|E(t - 1)^T E(t - 1)\|_F$ using Equation 4 and 5, respectively, and add to them the other terms in the equations to get new upper bounds of next step.
When a single update involves only one element of $A$, $\Delta(t)$ has a single non-zero element. Let $\Delta_{ij}(t)$ be the non-zero element of $\Delta(t)$, then

$$\|A^T \Delta(t)\|_F = \Delta_{ij}(t)\|A_{ij}\|_2 \text{ and } \|E(t-1)^T \Delta(t)\|_F = \Delta_{ij}(t)\|E(t-1)_{is}\|_2$$

where $A_{is}$ and $E(t-1)_{is}$ are the $i$th row of $A$ and $E(t-1)$, respectively.

There are two ways to compute $\|A^T \Delta(t)\|_F$ or $\|E(t-1)^T \Delta(t)\|_F$: (1) keep the matrix $E(t-1)$ and use Equation 6; (2) use the maximum element of $A$ or $E(t-1)$ to estimate. (1) is accurate and involves $O(n)$ operations. (2) is fast (only scalar multiplication). The actual choice depends on application.

When an update changes a row of $A$, computing $\|A^T \Delta(t)\|_F$ and $\|E(t-1)^T \Delta(t)\|_F$ is more expensive and requires $O(n^2)$ operations and $\|\Delta(t)^T \Delta(t)\|_F = \|\Delta_{is}(t)\|_2^n$ which is $O(n)$. This is at the same level of complexity as HITS but can be substantially cheaper to run because the latter takes a number of iterations to converge while the former needs to run only once. Kleinberg reported that the typical number of iterations for HITS to converge is 20 [6]. If the cost is still too high to accept, there are two ways to alleviate: (1) Frobenius norm has the property $\|AB\|_F \leq \|A\|_F \|B\|_F$. $\|A^T \Delta(t)\|_F$ and $\|E(t-1)^T \Delta(t)\|_F$ can be reduced to scaler multiplication (with loss of “tightness”); (2) the computation can be made to be distributed across all clients, as described in Section 6.

## 5 Evaluation

Compared to HITS, OnlineHITS is at the same complexity level. However, its advantage lies in the hope that the updates may not cause too much perturbation to the ranking so that recomputation is avoided. In addition, the operations introduced for perturbation checking do not require iteration so they are substantially cheaper than HITS. The benefit gained by Online HITS depends on the stability of the system in face of perturbation, which is application-specific. We believe that in situations where data is accumulating, Online HITS is most likely advantageous. The intuition behind this belief is that the more data is accumulated, the less significant a new update would be to the overall ranking. Therefore there would be more opportunities to avoid update and running of HITS.

To evaluate how well Online HITS performs, we run it on simulated data. Note that we are not testing how well the ranking produced by Online HITS (or HITS) fits the “real” ranking which is a rather qualitative and subjective measure. Instead, we are examining its algorithmic properties and how it performs better than original HITS.

Our test simulates a system of 4 users and 8 documents. A log item encoding a random user accessing a random document is generated at certain interval. The Online HITS constantly monitors the log and performs operations as described in Section 4. A total of 13547 log items are generated. We use a simple weight function: $w(i,j) =$ the number of times user $i$ accesses document $j$. The precision is chosen to be $\epsilon = 0.1$. Results are plotted in Figure 1 to 3.

Figure 1 shows, for each update, the actual perturbation $\|E_u\|_F$, the upper bound we estimated based on the method of Section 4.3, and the tolerance as specified by Theorem 1. This figure shows that the tolerance grows as the data accumulates and allows for more and more perturbation while maintaining the precision. Two areas have been enlarged to show the details of the plot. The first enlarged area illustrates the cross point (around 350) where the tolerance becomes greater than perturbation. This is where OnlineHITS starts saving. The second enlarged area lies between data item 11625 to 11675 and shows the situation when the amount of data accumulated is large. The horizontal line segment of dots shows the interval where the perturbation is within tolerable range and no update is applied. This particular line demonstrates around 30 updates for which the NewHITS needs not to be invoked, i.e., a saving of 30 rounds of HITS computation.

Figure 2 plots the ratio of the estimated upper bound of $\|E_u\|_F$ and its actual value. It shows how tight the upper bound given in Section 4.3 is. The number varies as updates accumulate and are applied, but never exceeds 3.2.
Figure 1: Accumulated perturbation $\|E_s\|_F$ and tolerance.

Figure 2: Approximation ratio.

Document rankings are plotted in Figure 3 which shows both the actual ranking of each document (obtained by running HITS at each update) and the approximation produced by OnlineHITS. Again part of the graph is enlarged for clarity. It is clear that the approximation is always within $\epsilon$.

Our simulation demonstrates the substantial advantage of OnlineHITS, especially when the amount of data accumulated is large. And this confirms our intuition that the more data is accumulated, the less significant a new update would be to the overall ranking.

6 Privacy Preserving Online HITS

The algorithm described in previous sections addresses the dynamic and real-time response issues of using access patterns in link analysis. However, in many situations, a naive implementation of the algorithm has severe privacy implications. In most applications, the weight on each edge of the graph...
represents the “rating” or “preference” of a user to the documents and is gathered via client side logging. Such document access information is quite personal and exposing it would jeopardize the privacy of users thus hindering the acceptance of our system. If implemented directly, the online HITS algorithm would require the server running the algorithm be able to see all the data and involve substantial amount of network communication. In most situations trusting the server or network is not acceptable.

Fortunately this problem can be solved with cryptographic techniques. The general idea is that we can use encryption to protect user data and perform computations on encrypted data. Only aggregate data is made public and individual data are transmitted across the network in encrypted form. Such scheme has been proven to be sound and feasible, with satisfying performance. In [15], Canny proposed a privacy preserving collaborative filtering scheme that performs iterative SVD using the conjugate gradient method of Polak-Ribiere [16]. The basic building blocks, however, are homomorphism and threshold decryption which allow one to compute sums without disclosing summands. And this is exactly all that is needed to perform Online HITS. [15] gives complete description of the scheme and the proof of its soundness. Here we will sketch how it can be applied to make Online HITS distributed and privacy preserving.

In the following we will only consider the computation of document ranking, \( x \). Expertise ranking is done in a similar fashion. We assume that most clients are honest and won’t cheat or collude to pry about other’s data. As in [15], we assume there is a write-once read-many (WORM) storage system where public data can be published. We also assume there is a tallier machine that performs addition on the data it receives. The tallier doesn’t have to be a dedicated server. One of the clients can be designated as the tallier or its task can be made peer-to-peer.

6.1 Basics

Several commonly used encryption schemes (RSA, Diffie-Hellman, ECC) have a useful property called homomorphism: if \( m_i \) is a message and \( E(\cdot) \) is an encryption function, let \( g \) be a multiplicative group element, we can define a function \( H(m) = E(g^m) \). This function satisfies

\[
H(\sum_{i=1}^{n} m_i) = \prod_{i=1}^{n} H(m_i)
\]
where multiplication is ring multiplication for RSA, or element-wise for DH or ECC. This allows one to obtain the encryption of a sum without revealing the summands.

Recovering the sum involves secret sharing and threshold decryption. The decryption key is not owned by any single party but secret-shared among all the clients. Pedersen’s key generation protocol [17] or its variants/enhancements [18, 19] can be used to securely generate the globally-known public key and distribute the secret shares of the private key among participants. At the end of their protocol, each client will have a share $s_i$ of the decryption key, $s$, which could have been easily reconstructed from any set $\Lambda$ of $t + 1$ shares via Lagrange interpolation where $t$ is a pre-defined threshold that is greater than the maximum number of dishonest clients in the system. This arrangement not only discourages clients from cheating but also introduces redundancy that makes the system robust – any $t + 1$ shares of $s$ can recover it. However, reconstructing $s$ will effectively reveal the secret key to a single party thus rendering the scheme useless. Instead, we use threshold decryption on the ciphertext when decryption is desired. That is, each client decrypts the ciphertext with its share of the key and the result is a share of the decryption of the value. By putting these shares together, users can recover the encrypted value.

The value decrypted is not actually the sum of messages $\sum_{i=1}^{n} m_i$ that we are seeking, rather it is $g^{\sum_{i=1}^{n} m_i}$. Although recovering the sum requires taking discrete log, the value of sum will be small enough ($10^6$ to $10^9$) so that a baby-step/giant-step approach is practical and the process can also be sped up by distributing it among many clients to be performed in parallel.

6.2 A Run of HITS

The results of the $t$th iteration of HITS, $x^t$ and $y^t$ which are aggregate data, are made public. User $i$ is responsible for his own rating of the documents (obtained via analyzing his document access pattern), namely the $i$th row of matrix $A$. Let $A^T_i = [a_{i1}, a_{i2}, \ldots, a_{im}]$ be that row. For the step $x^{t+1} = A^T y^t$, all that is involved from $i$ is his own expertise ranking, $y^t_i$, and $A^T_i$. User $i$ computes $y^t_i A^T_i$ and encrypts the vector and sends it to the tallier. The tallier, after receiving data from all users, produces the encryption of the sums of the ranking of each document by multiplying corresponding elements of the vectors. The resulting vectors are committed to WORM. Users will read from WORM and perform threshold decryption to recover the values. This is $x^{t+1}$.

To compute $y^{t+1}$ (which is $Ax^{t+1}$), user $i$ computes $A^T_i x^{t+1}$ which is $y^{t+1}_i$, the $i$th element of vector $y$ at iteration $t$, and publish it. If every user does this, the vector $y^{t+1}$ can be obtained.

The iteration can stop when enough precision is achieved.

6.3 Perturbation Checking

The scheme described in Section 6.2 shows a run of HITS, not Online HITS. To fit online HITS into such a scheme, we need to find a way to compute the perturbation, $\|E_S(t)\|_F$, with encrypted data or allow each user to compute with local data.

Recall that Equations 3, 4, 5 and 6 give us a way to update the upper bound of $\|E_S(t)\|_F$. The terms that need to be computed for each update are $\|A^T \Delta(t)\|_F$, $\|E(t-1)^T \Delta(t)\|_F$ and $\|\Delta(t)^T \Delta(t)\|_F$. Since for user $i$, $\Delta(t)$ has non-zero elements only in its $i$th row (and these numbers are obtained locally via his document access pattern analysis), $A^T \Delta(t)$ only involves the $i$th row of $A$, which the user maintains. Similarly, $E(t-1)^T \Delta(t)$ only involves the $i$th row of $E(t-1)$. In short, each user’s update only involves his local data and it is straightforward to perform perturbation checking without disclosing private data: $\|E_S(t)\|_F$, $\|A^T E(t)\|_F$ and $\|E(t)^T E(t)\|_F$ are made public via the WORM storage and each user will update them using Equations 4, 5 and 6 with their local updates. When it is determined that it is time to update $A$, each user will update his own row and reset the perturbation records. All of them then collaboratively run the HITS algorithm as described in Section 6.2.

Note that we have actually killed two birds with one stone if we perform perturbation checking this
way. Not only could we preserve user’s privacy, we also distributed the computation among all users and parallelized the process.

There are some other issues in making such a scheme realistic such as dealing with dishonest users/tallier. Addressing them is out of the scope of this paper. [15] discusses such issues in detail and gives feasible solutions. In particular, it uses Zero Knowledge Proof (ZKP) to validate the data user and tallier produces so that they cannot excessively influence the results by cheating on their values.

7 Related Work

In [1], a set of heuristic graph algorithms are used to uncover shared-interest relationships among people, and to discover the individuals with particular interests and expertise, based on the logs of email communication between these individuals. The limitation with this approach is that experts are assumed to be communicating with fellow experts, which is not necessarily true in the real-world where experts may not be acquainted with one another, or may be rivals. Our approach does not assume any particular communication patterns between experts, and instead locate the experts based on their activities, e.g. if an expert accesses this set of authoritative documents, another person who accesses the same set is likely to be an expert as well.

The Referral Web [3, 4] is an interactive system for restructuring, visualizing and searching social networks on the World Wide Web. It constructs a graph of all users based on their email communication logs, which it uses to send a chain of referral requests until these requests reach an expert user. Like our Online HITS algorithm, Referral Web constructs the social network incrementally as it indexes the documents created and received by users. In contrast to our approach, however, the Referral Web raises possible privacy concerns because the chain of referrals inevitably reveal who someone down the chain knows to the user who initiates the search, unless individuals down the chain chooses not to forward the referral, in which case it becomes harder for the query to succeed.

Pirolli et al. [20] use a link-based approach like HITS to categorize webpages. It is similar to our weight-based algorithm in that users’ access paths and metadata about webpages are used to construct the appropriate matrices. It differs significantly from ours in that while we use successive iterations to converge on our results, Pirolli et al. construct an activation network based on the strength of association between webpages and use the spread of activation in this network, starting from identified source webpages, to identify the webpages that exceed a threshold quantity of flow.

Carriere and Kazman’s WebQuery system [21] rank webpages by considering the number of neighbors in the hyperlink structure that each webpage has. WebQuery performs link-based query post-processing to improve the quality of the results that it returns. In contrast, our incremental approach assumes that the hyperlink structure is highly dynamic, and postpones processing until the latest user-document accesses accumulate significant perturbation.

8 Conclusion

We extended the HITS hyperlink analysis algorithm to make it applicable to analyzing user-document factor graph constructed from user-document access patterns. Our generalizations are in two directions. First, we replaced the 0-1 valued hyper-link property to a non-negative valued weight function to model the users’ access behavior more accurately and proved its convergence.

Second, we created an online eigenvector calculation method that can compute the results of mutual reinforcement voting efficiently in face of frequent updates by estimating the upper bound of perturbation and postponing applying the updates whenever possible. Both theoretical analysis and empirical experiments show that our generalized online algorithm is more efficient than the original HITS under the context of dynamic data.
Finally we developed a secure variation of our online algorithm that solves the potential privacy issues that may occur when deploying large-scale access pattern-based document and authority ranking systems. Our scheme makes use of cryptographic techniques such as threshold decryption and homomorphism and distributes computation among users. Only aggregate or encrypted data are exposed. The scheme is also robust against a number of dishonest users up to a certain threshold.

Our extensions to Kleinberg’s original HITS algorithm result in a generalized algorithm, Secure OnlineHITS, that is practical for link analysis in scenarios such as collaborative work and online communities, in which there is no explicit link structure among nodes, and that users’ access patterns of documents are highly dynamic, complex and should remain private.

References


A Proof of Theorem 1

Proof We use $\tilde{}$ to represent perturbed quantity. Suppose $S \in \mathbf{R}^{n \times n}$ is a symmetric matrix with principal eigenpair $(\lambda^*, a^*)$, and eigengap $\delta > 0$. Let $E_s$ be a symmetric perturbation to $S$ such that $\tilde{S} = S + E_s$. By Theorem V.2.8 from matrix perturbation theory[22], there is some eigenpair of $\tilde{S} (\tilde{\lambda}, \tilde{a})$ such that

$$\|a^* - \tilde{a}\|_F \leq \frac{4\|E_s\|_F}{\delta - \sqrt{2}\|E_s\|_F}$$

and

$$|\lambda^* - \tilde{\lambda}| \leq \sqrt{2}\|E_s\|_F$$

assuming the denominator in 7 is positive. Let $L \in \mathbf{R}^{n-1 \times n-1}$ be diagonal matrix containing all $S$’s eigenvalues except $\lambda^*$. A bound similar to 8 holds:

$$\|L - \tilde{L}\|_F \leq \sqrt{2}\|E_s\|_F$$

Let $\tilde{\lambda}_2$ be the largest eigenvalue in $\tilde{L}$. By Corollary IV.3.6 of [22], Equation 9 implies

$$\tilde{\lambda}_2 \leq \lambda_2 + \sqrt{2}\|E_s\|_F$$

Since $\|E_s\|_F \leq \frac{\delta}{2\sqrt{2}}$, Equations 8 and 10 ensures that $\tilde{\lambda} > \tilde{\lambda}_2$, i.e. $(\tilde{\lambda}, \tilde{a})$ is indeed the principal eigenpair of $\tilde{S}$. Also this will ensure the denominator in 7 is indeed positive.

Given any $\epsilon > 0$, if $\|E_s\|_F \leq \frac{\epsilon}{4\sqrt{2}}$, then $\frac{4\|E_s\|_F}{\delta - \sqrt{2}\|E_s\|_F} \leq \epsilon$ thus we have $\|a^* - \tilde{a}\|_F \leq \epsilon$