A mathematical approach to scalable personalized PageRank

Rest, F.W. van

Frank van Rest

A mathematical approach to scalable personalized PageRank

Bachelor thesis, May 20, 2009

Thesis advisor: Dr. F.M. Spieksma

Mathematisch Instituut, Universiteit Leiden

Contents

1 PageRank

1.1 Introduction

PageRank is developed at Stanford University by the two PhD-students Sergey Brin and Larry Page and first introduced in [18]. Its purpose was to solve the problem of the growing Web[1]. Search engines used to return a lot of results at a given search term, but could not provide a useful order for this results. Because of the exponentially growing Web, the usability of these results was getting worse and worse. PageRank is a web ranking system. Its ranking is based on the link structure of the Web. The main idea is that webpages are important, if linked to by important webpages.

PageRank is implemented by Google for its search results. PageRank is often stated as the most important factor in Google's scoring. This is not - at least not any more [16]- the case. It is nevertheless clear that knowledge about PageRank is useful in other areas, f.e. to determine which webpages to crawl (see chapter 7.4).

Calculating PageRank is quite time and memory consuming. This cannot be done online at the moment a user is searching (query time), because it takes weeks to calculate, and the user wants an answer within several seconds or even less. Fortunately, the PageRank score is independent of the search queried by the user. Therefore it is possible to calculate the PageRank scores for all webpages offline. The PageRank scores are saved in memory and can be used whenever a user searches online.

The PageRank that is described in [18] gives a universal score for the pages of the web. It is however possible to change the calculations so that the results will reflect someone's personal preferences. One could think that these personalized calculations could make personalized PageRanks for all users, thereby improving the personal search experiences. However the calculations are requiring so many resources, that this is not possible to do for all users.

In this thesis we describe other - smarter - ways to personalize PageRanks. We also describe some experiments and give some general acknowledgements. In the section 7 we comment on some articles in this field, f.e. on [13], for which we did quite some work to clarify.

[1] Web stands for World Wide Web

1.2 Notation

For the purpose of general understanding we first introduce the notation in this thesis.

All vectors in this thesis are recognizable by their overline and will be expressed by row vectors (e.g. $\overline{u}$ is a row vector, which makes $\overline{u}^T$ a column vector). For all vectors in this thesis we have

$$||\overline{v}|| \ = \sum_i |v(i)| \leq 1.$$

The vector should be seen as distributions rather than arrows. The i'th component of a vector is denoted by $v(i)$. Scalars are in lower case (e.g. c). Matrices will be written in upper case (e.g. A). Matrices are always of size n^2, where n is the size of the web. Matrices are often (sub-)stochastic, but can be binary. The element in the i'th row and j'th column is denoted by A_{ij}.

$\overline{e}$ is the vector consisting of ones only, often used to sum rows or columns or to find a norm:

$$\overline{e} = [1, 1, \ldots, 1, 1] \, .$$

$\overline{e_i}$ is the i'th unit vector.

In this thesis the words *webpage* and *node* are synonyms just like the words *link* and *arc*.

1.3 Web as a directed graph

The web can be seen as a directed graph $(V, \tilde{A})$. Each webpage is a node $v \in V$ and a link from webpage v_1 to webpage v_2 is an arc $a \in \tilde{A}$. This graph can be represented by its adjacency (binary) matrix B

$$B_{ij} = \begin{cases} 1 & \text{if } v_i \text{ links to } v_j, \\ 0 & \text{elsewhere.} \end{cases}$$

Now

$$\overline{O}^T = B\overline{e}^T$$

is the vector with components equal to the number of outgoing links for all webpages (e.g. $O(i)$ is the number of outgoing links on webpage i). We also have

$$\overline{I} = \overline{e}B$$

the vector with components equal to the number of incoming links for all webpages (e.g. $I(i)$ is the number of incoming links for webpage i).

We call nodes $\{i : O(i) = 0\} \subset V$ dangling nodes (in words: webpages without outgoing links). Now

$$C_{ij} = \begin{cases} \frac{1}{O(i)} & \text{if } O(i) \neq 0 \text{ (or } i \text{ is no dangling node),} \\ 0 & \text{elsewhere} \end{cases}$$

is a sub-stochastic matrix (rowsums is 1 or 0).

To create a stochastic matrix we alter C by introducing a distribution vector $\overline{w} : ||\overline{w}|| = 1$.

$$A_{ij} = \begin{cases} \frac{1}{O(i)} & \text{if } O(i) \neq 0 \text{ (or } i \text{ is no dangling node),} \\ w(j) & i \text{ is a dangling node.} \end{cases}$$

In terms of the web this can be seen as altering the web so that all dangling nodes are connected to the nodes for which the corresponding element in $\overline{w}$ is nonzero. Now A is a stochastic matrix. The right eigenvector is $\overline{e}^T$ with eigenvalue 1:

$$A\overline{e}^T = \overline{e}^T.$$

This implies that there exists also at least 1 left eigenvector corresponding to eigenvalue 1.

1.4 Random Walk on the Web

On this web (or in this matrix) we can observe a random walk. Start in any node v_i of the graph and walk to v_j with probability A_{ij}. Now the distribution of the position in the long run yields information about which nodes can be viewed as more *important* then others.

To calculate this distribution we can use Markov theory. In Markov theory this distribution is called the stationary distribution of a Markov chain. Here A is the transition matrix of the Markov chain and the stationary distribution satisfies

$$\overline{\pi} = \overline{\pi}A,$$

where $\overline{\pi}$ is the left eigenvector corresponding to eigenvalue 1. This $\overline{\pi}$ is not necessary unique, but is unique if the Markov chain is irreducible and aperiodic. Therefor we alter A one more time.

First we introduce a so called teleportation vector $\overline{u} : ||\overline{u}|| = 1$. This vector describes a uniform distribution on all webpages. Note that $\overline{u}$ can be equal to $\overline{w}$. We also introduce a damping factor c (in [18] c is chosen 0.85).

$$P = cA + (1 - c)\overline{e}^T\overline{u}.$$

One can interpret this modification as follows: the random walker has probability $(1 - c)$ at every step to be teleported to a random page of the web, independent of the outgoing links of the webpage of its position. It is clear that this modification makes P irreducible and aperiodic: all nodes are connected.

PageRank is the unique vector $\overline{\pi}$ with

$$\overline{\pi} = \overline{\pi}P.$$

Because P is aperiodic and irreducible (or ergodic), this stationary distribution $\overline{\pi}$ can be found by the powermethod:

$$\overline{\pi}^1 = \overline{\pi}^0 P$$

$$\overline{\pi}^2 = \overline{\pi}^1 P$$

$$\cdots$$

$$\overline{\pi}^N = \overline{\pi}^{N-1} P$$

with $N \to \infty$, for any probability vector $\overline{\pi}^0$.

Applying this gives

$$\overline{\pi}^N = \overline{\pi}^0 P^{N-1}.$$

Because P is irreducibel and aperiodic,

$$\lim_{n\to\infty} P^n = \begin{pmatrix} \pi \\ \pi \\ \vdots \\ \pi \\ \pi \end{pmatrix}.$$

$\overline{\pi}^0 P^n \to \overline{\pi}$ for all initial probability vectors $\overline{\pi}^0$.

This method does not find a approximation, but is able to find the exact solution to a certain precision.

If N denotes the number of steps then we can approximate $\overline{\pi}$ by precision $\epsilon = (1-c)(c^{N+1} + c^{N+2} + \ldots) = c^{N+1}$. This method is described in [18] as a very fast way to calculate a satisfactory PageRank $\overline{\pi}$.

1.5 Alternative notation

Now the PageRank formula can be described as

$$\begin{aligned} \overline{\pi} &= \overline{\pi}P \\ &= \overline{\pi}(cA + (1-c)\overline{e}^T\overline{u}) \\ &= \overline{\pi}cA + (1-c)\overline{\pi}\overline{e}^T\overline{u} \\ &= \overline{\pi}cA + (1-c)\overline{u}, \end{aligned} \tag{1}$$

where we have used $\overline{\pi}^T\overline{e} = 1$ since $||\overline{\pi}|| = 1$ in the last equation. Hence,

$$\overline{\pi}(I - cA) = (1-c)\overline{u} \tag{2}$$

and so

$$\begin{aligned} \overline{\pi} &= (1-c)\overline{u}(I - cA)^{-1} \\ &= (1-c)\overline{u}\sum_{n=0}^{\infty} c^n A^n. \end{aligned} \tag{3}$$

In this notation, it is easier to see that the PageRank vector defines a probability: the probability of being in node i during a random walk of N steps with probability $(1-c)c^N$ over the graph defined by A, when started in an initial node according to probability distribution $\overline{u}$.

Note that the formula for $\overline{\pi}$ in equation 3 does not provide a good way to calculate $\overline{\pi}$. Calculating A^n in the summation involves the multiplication of two n^2 matrices. Matrix multiplication is too complex to be done with matrices of size $n > 1$ trillion, which [2] claimed as the size of the web.

2 Personalization of PageRank

In this section we will discuss a property of PageRank and we will prove a theorem on convex combinations. Together these create insight in how to combine specific PageRanks, so that they can be used to create personalized PageRanks.

Personalizing search results improves user experience by choosing another (than default) order of the search results.

Example 2.1. *Person A is basketball fan and likes to read the latest transfer details of players from the Chicago Bulls (the basketball team from Chicago, USA). He will use the query "bulls prices". Person B is farmer and wants to have up to date information about the value of his stock. He will also use the query "bulls prices". Without personalizing, the search engine has to give the same results to both persons. At least one of the persons will not have the best search experience possible. When the search engine has knowledge of the background of the user, it will be able to serve the user with the right personalized results, thereby providing the user a better search experience.*

(Of course this can be influenced by inserting extra search terms).

2.1 Teleportation vector $\overline{u}$ and Point of View PageRank

In the PageRank formula the teleportation vector $\overline{u}$ has a great influence on determining the outcome. The biasing of the PageRank by this vector is first suggested in [17], and thereafter explored many times ([9],[10],[11]). In this thesis the goal of biasing the PageRank with the teleportation vector is to personalize the search results. However, it may also be used for other purposes, f.e. to fight spam in the search results by giving spam pages probability 0 in $\overline{u}$.

By changing the teleportation probabilities $\overline{u}$ we can personalize PageRank in quite an intuitive way: by increasing the probability of teleporting to a page in the user's field of interest and by decreasing the probability of teleporting to a page outside the user's field of interest.

In theory one could provide one's personal $\overline{u}$ (think of a uniform distribution over the pages one has bookmarked). It is nevertheless impossible to calculate a personalized pagerank for every user, because the calculations are requiring so many resources.

One of the possibilities is to take $\overline{u} = \overline{e_i}$. This gives a special PageRank $\overline{\pi_i}$. One can look at this PageRank as the ranking of the web from page i's point of view. In the light of the random walk, this special model describes a random walk starting on page i, for N steps.

Under π_i, pages in the 'neighborhood' of page i will have a relative high PageRank. Pages are in the 'neighborhood' of i when the number of steps to get from i to the page is relative small. The web is a small world [1], so relative small should be thought of as in 1, 2 maybe 3 steps.

2.2 Convex Combinations of PageRanks

Theorem 2.1. *For any convex combination of two teleportation vectors u_1, u_2:*

$$\lambda \overline{u}_1 + (1-\lambda)\overline{u}_2 = \overline{u}_3$$

we have

$$\lambda \overline{\pi}_{u_1} + (1-\lambda)\overline{\pi}_{u_2} = \overline{\pi}_{u_3}.$$

Proof.

$$\begin{aligned}
\lambda \overline{\pi}_{u_1} + (1-\lambda)\overline{\pi}_{u_2} &= \lambda(1-c)\overline{u}_1 \sum_{n=0}^{\infty} c^n A^n + (1-\lambda)(1-c)\overline{u_2} \sum_{n=0}^{\infty} c^n A^n \\
&= (\lambda \overline{u}_1 + (1-\lambda)\overline{u}_2)(1-c) \sum_{n=0}^{\infty} c^n A^n \\
&= \overline{u}_3 (1-c) \sum_{n=0}^{\infty} c^n A^n.
\end{aligned}$$

□

So a convex combination of teleportation vectors will lead to the same convex combination of PageRank vectors. This result is true for all PageRank combinations, but of special value for Point of View PageRank. This result makes it easy to generate new PageRanks from existing Point of View PageRanks. We believe the proof is more transparent than the proof given in [13], due to representation (3) of $\overline{\pi}$.

2.3 Personalization in N dimensions

As stated in [13] these results can be used to introduce personalization to a certain level. One can calculate the Point of View PageRank for a number of webpages, $\pi_1, \ldots, \pi_N$. One can than define a convex combination for each user, $\lambda_1, \ldots, \lambda_N \sum_{i=1}^{N} \lambda_i = 1$, which doesn't need any calculations per se and is very easy (small) to store. During query time a Personalized PageRank can be calculated simply by applying theorem 2.1 on the user defined λ_i's and the precalculated π_i's.

Complexity for calculating the π_i's is growing linear with N. In [13] is stated how decomposition of A can be used to decrease complexity for bigger N.

Example 2.2. *The Open Directory Project is a directory of links maintained by volunteers. It exists of 16 topics (e.g. Arts, Business and Science). Each topic has a webpage that links to webpages of subtopics and webpages that are relevant for the topic.*

One can calculate the Point of View PageRank for each webpage of these 16 topics. By letting users score their interest in each topic, one can create a Personalized PageRank based on these Point of View PageRanks.

This is the example given for personalization in N dimensions in [9],[10] and [11].

3 Decomposition of A using a taboo set

Random walks can be observed in different ways. One such way is to decompose the walks in different smaller parts. We use the last exit decomposition described in [7] and first applied to PageRank in [13].

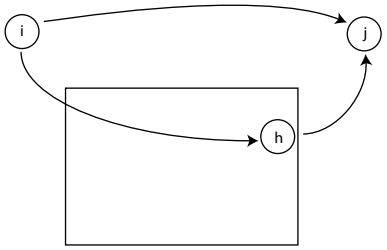

Figure 1: Last exit decomposition with the rectangle H

First we look at all the walks that do not pass any state in a subset[2] H. These walks run on the matrix

$$ {}_H A_{ij} = \begin{cases} 0 & j \in H, \\ A_{ij} & \text{else.} \end{cases} $$

$$ ({}_H A)^0 = I. $$

All others walks will pass subset H. You can decompose each of such 'passing H'-walks into two segments. The first segment of the walks passing H consists of the part where H may be visited, up to the time of the last visit. The last segment starts in $h \in H$ and consists of the walk that does not visit H anymore. This is called the last exit decomposition.

For the last exit decomposition of A over a subset $H \subset V$, we get

$$ A^n_{ij} = \sum_{l=1}^{n-1} \sum_{h \in H} A^l_{ih} ({}_H A)^{n-l}_{hj} + ({}_H A)^n_{ij} $$

for $i, j \notin H$.

In the next section we apply this decomposition to the Point of View Pagerank.

[2] also known as a taboo set

3.1 Point of View PageRank

The last-exit-decomposition can be applied to the Point of View PageRank:

$$\begin{aligned}\overline{\pi_i} &= (1-c)\overline{e_i}\sum_{n=0}^{\infty} c^n A^n \\ &= (1-c)\overline{e_i}I + (1-c)\overline{e_i}\sum_{n=1}^{\infty} c^n A^n.\end{aligned}$$

Hence for $i, j \notin H$,

$$\begin{aligned}\pi_i(j) &= (1-c)e_i(j) + (1-c)e_i(i)\sum_{n=1}^{\infty} c^n A_{ij}^n \\ &= (1-c)e_i(j) \\ &\quad +(1-c)e_i(i)\sum_{n=1}^{\infty} c^n (\sum_{l=1}^{n-1}\sum_{h\in H} A_{ih}^l ({}_H A)_{hj}^{n-l} + ({}_H A)_{ij}^n) \qquad (4) \\ &= (1-c)e_i(j) + (1-c)e_i(i)\sum_{n=1}^{\infty} c^n ({}_H A)_{ij}^n \\ &\quad +(1-c)e_i(i)\sum_{n=1}^{\infty} c^n (\sum_{l=1}^{n-1}\sum_{h\in H} A_{ih}^l ({}_H A)_{hj}^{n-l}),\end{aligned}$$

where the empty sum $\sum_{l=1}^{0}$ in equality (4) equals zero. We have for

$$\begin{aligned}&(1-c)e_i(i)\sum_{n=1}^{\infty} c^n (\sum_{l=1}^{n-1}\sum_{h\in H} A_{ih}^l ({}_H A)_{hj}^{n-l}) \\ &\quad = (1-c)e_i(i)\sum_{h\in H}\sum_{l=1}^{\infty} A_{ih}^l c^l \sum_{n=l+1}^{\infty} ({}_H A)_{hj}^{n-l} c^{n-l} \qquad (5) \\ &\quad = (1-c)e_i(i)\sum_{h\in H}\sum_{l=1}^{\infty} A_{ih}^l c^l \sum_{n=1}^{\infty} ({}_H A)_{hj}^n c^n.\end{aligned}$$

We have used interchange of summation to obtain equality (5):

$$\sum_{n=1}^{\infty}\sum_{l=1}^{n-1} = \sum_{l=1}^{\infty}\sum_{n=l+1}^{\infty}.$$

So:

$$\begin{aligned}\pi_i(j) = (1-c)e_i(j) + (1-c)e_i(i)\sum_{n=1}^{\infty} c^n ({}_H A)_{ij}^n \\ +(1-c)e_i(i)\sum_{h\in H}\sum_{l=1}^{\infty} A_{ih}^l c^l \sum_{n=1}^{\infty} ({}_H A)_{hj}^n c^n.\end{aligned}$$

3.2 Explanation

$\pi_i(j)$ is the PageRank value of page j from page i's Point of View. It consists of three parts:

$$(1-c)e_i(j)$$

equals $1-c$ when $i=j$ and 0 when $i \neq j$. This describes the random walk from i to itself by teleportation (in zero steps);

$$(1-c)e_i(i)\sum_{n=1}^{\infty} c^n ({}_H A)^n_{ij}$$

is the part of the equation standing for all random walks from i to j without passing any site in H (site i and j both not in H);

$$(1-c)e_i(i)\sum_{h\in H}\sum_{l=1}^{\infty} A^l_{ih}(1-c)^l \sum_{n=1}^{\infty}({}_H A)^n_{hj} c^n$$

is the part of the equation standing for all random walks from i to j that pass a site in H. In the sum over all $h \in H$ this product of two sums is written in a special form. The first sum is a new random walk from i to h (with walks consisting of at least one step) or in formula form: $\pi_i(h) - (1-c)e_i(h)$. Note that this formula is independent of endpoint j. The last sum is the random walk from h to the end-point j without passing H. Note that this is independent of starting point i.

Therefore, both parts of this equation can be calculated once for all $h \in H$, and reused for all i and for all j.

A shorter notation for $\pi_i(j)$ is therefore:

$$\pi_i(j) = (1-c)e_i(j) + (1-c)\sum_{n=1}^{\infty} c^n ({}_H A)^n_{ij} + \sum_{h\in H}(\pi_i(h) - (1-c)e_i(h))\sum_{n=1}^{\infty}({}_H A)^n_{hj} c^n.$$

In [13] is a proof of three pages added for the above result. It was part of this thesis research to make this more clear. We believe that using taboo theory and introducing a more formal notation, the result can be achieved in a more transparent manner.

4 Dangling nodes

Dangling nodes are webpages which do not have outgoing links. Without the solution proposed in 1.3 of restarting a new random walk, these nodes become probability sinks. This is an unwanted effect. Other than the addition of $\overline{w}$, there are several solutions for this problem.

In [6] is suggested to remove the dangling nodes before the calculations and add them after. It is not shown there how this should be done. In [14] is suggested to add the nodes for the final few iterations of the computations.

Another solution is lumping all dangling nodes into a single node [12].

Because the number of dangling nodes can be quite high (up to 50% of all nodes [12]), it may be very beneficial to find a solution that reduces the amount of work per dangling node.

In the next section we introduce solutions that reduces the number of nonzero elements by at least $(n-1) \times d$, where d is the number of dangling nodes. We do this by introducing an extra node to the graph. This solution is new and developed as a part of the research project for this thesis.

4.1 Extra Node Solution

We introduce two solutions for the dangling nodes. These solutions can be compared to the solution in subsection 1.3 with $\overline{w} = \overline{u}$.

Add another node v and link all dangling nodes to v. The outgoing link distribution of v is equal to the teleportation vector $\overline{u}$. By renumbering the nodes so that the dangling nodes have highest numbers, this can be presented as

$$P' = \left(\begin{array}{c|c} P_{nd} & 0 \\ \hline P_d & 1 \\ \hline \overline{u} & 0 \end{array}\right),$$

where P_{nd} $(n \times \#nd)$ are the rows corresponding to the non-dangling nodes and P_d $(n \times \#d)$ are the rows corresponding to the dangling nodes ($\#nd$ is the number of non-dangling nodes and $\#d$ is the number of dangling nodes).

A step further is to remove the teleportation from the rows itself, by using the sub-stochastic matrix A $(n \times n)$ and change it so that the teleportation is done by adding a link to the extra node with probability $(1 - c)$.

$$P'' = \left(\begin{array}{c|c} cA_{nd} & (1-c) \\ \hline A_d & 1 \\ \hline \overline{u} & 0 \end{array}\right).$$

In the next subsection we will prove that this does not change the order of the PageRanks of the legitimate webpages, compared to the solution in subsection 1.3 with $\overline{w} = \overline{u}$.

4.2 Proof of Extra Node Solution

It is not straightforward that the Extra Node Solution is indeed correct and does not affect the order of the PageRanks of the legitimate webpages.

The proof comes from general Markov Chain theory. We define $f_{ij}^{(n)}$ the first entrance probability of j starting in i at time n. Now

$$f_{ij}^{*} = \sum_{n=1}^{\infty} f_{ij}^{(n)}$$

is the probability that that a random walk will reach j at least once starting in i. Further,

$$m_{ij} = \sum_{n=1}^{\infty} n f_{ij}^{(n)}$$

is the mean first entrance time of j. m_{ii} is the mean recurrence time of i. The stationary probability can be described as

$$\pi(i) = \frac{f_{ii}^{*}}{m_{ii}} = \frac{1}{m_{ii}}.$$

By [7] we know that

$$\frac{\pi(j)}{\pi(i)} = \frac{m_{ii}}{m_{jj}} = \sum_{n=0}^{\infty} {}_{i}p_{ij}^{(n)} =: {}_{i}p_{ij}^{*}, \tag{6}$$

where ${}_{i}p_{ij}^{*}$ can be interpreted as the expected number of visits of j before returning to i.

Now we compare probabilities p and $\pi(i)$ in the original web, with the probabilities $\tilde{p}$ and $\tilde{\pi}(i)$ in the web with the extra node.

$${}_{i}p_{ij}^{*} = {}_{i}\tilde{p}_{ij}^{*}, \quad i, j \neq v.$$

The extra node does not change the probability ${}_{i}p_{ij}^{*}$ for any $i, j \neq v$. So the proportion remains the same for any two pages. This implies

$$\frac{\pi(j)}{\pi(i)} = \frac{\tilde{\pi}(j)}{\tilde{\pi}(i)}$$

$$\frac{\sum_{j \neq v} \pi(j)}{\pi(i)} = \frac{\sum_{j \neq v} \tilde{\pi}(j)}{\tilde{\pi}(i)}$$

$$\frac{1}{\pi(i)} = \frac{1 - \tilde{\pi}(v)}{\tilde{\pi}(i)}.$$

So

$$\pi(i)(1-\tilde{\pi}(v)) = \tilde{\pi}(i).$$

The absolute PageRank for any page does change, because some probability weight will remain in the extra node. It is important to keep this in mind, when using an algorithm resulting in convergence to a certain precision. The next section will provide insight in the amount of PageRank that remains in the extra node.

In [12] is suggested to lump all dangling nodes before calculations. Also a proof that this does not change the order of the PageRanks of the legitimate webpages is given. The construction in the proof above can also be used to proof the legitimacy of lumping all dangling nodes. Simply by lumping the dangling nodes so that

$${}_i p^*_{ij} = {}_i \tilde{p}^*_{ij}$$

with i, j non-dangling nodes. This can be done by using the same distribution for outgoing links in the dangling nodes as in the lumped node. We believe the proof is more transparent than the proof given in [12], due to use of taboo theory.

The taboo decomposition in equation 6 can be interpreted as

$$\pi(j) = \pi(i)_i p^*_{ij}$$

and for $j \notin A$:

$$\pi(j) = \sum_{a \in A} \pi(a)_A p^*_{aj},$$

where ${}_A p^*_{aj}$ can be interpreted analog as ${}_i p^*_{ij}$. By applying this substitution again for

$$\pi(a) = \sum_{b \in B} \pi(b)_B p^*_{ba},$$

we get for $j \notin A$:

$$\pi(j) = \sum_{a \in A} \sum_{b \in B} \pi(b)_B p^*_{ba} {}_A p^*_{aj}.$$

In section 5.2 we introduce an algorithm based on this double taboo decomposition.

Note that this taboo decomposition is different than the last exit decomposition described in section 3.

4.3 Weight in Extra Node

We have

$$\overline{\pi}P = \overline{\pi}$$
$$\tilde{\overline{\pi}}\tilde{P} = \tilde{\overline{\pi}}.$$

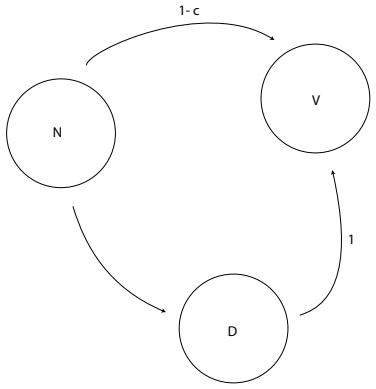

Figure 2: Weight in extra node v

Let d be the set of dangling nodes and let nd be the set of non-dangling nodes. Let $\sum_{j\in d}\tilde{\pi}(j) := \tilde{\pi}_d$ and $\sum_{j\in nd}\tilde{\pi}(j) := \tilde{\pi}_{nd}$ (with the analogon for π).

$$\begin{aligned}\tilde{\pi}(v) &= \tilde{\pi}_d + \tilde{\pi}_{nd}(1-c)\\ &= (1-\tilde{\pi}(v))\pi_d + \pi_{nd}(1-\tilde{\pi}(v))(1-c)\\ &= 1-\tilde{\pi}(v) - c\pi_{nd}(1-\tilde{\pi}(v)).\end{aligned}$$

The last equation is derived because $\pi_d = 1-\pi_{nd}$. This also implies that

$$\begin{aligned}\tilde{\pi}(v) &= 1-\tilde{\pi}(v) - c(1-\pi_d)(1-\tilde{\pi}(v))\\ &= 1-\tilde{\pi}(v) - c + c\tilde{\pi}(v) + c\pi_d(1-\tilde{\pi}(v)) \quad (7)\\ &= 1-\tilde{\pi}(v) - c + c\tilde{\pi}(v) + c\tilde{\pi}_d. \quad (8)\end{aligned}$$

For equation 7 we have

$$\tilde{\pi}(v)(2-c+c\pi_d) = 1-c+c\pi_d$$
$$\tilde{\pi}(v) = \frac{1-c+c\pi_d}{2-c+c\pi_d}.$$

So $\tilde{\pi}(v)$ is determined by π_d (and c, but c can be chosen).

We are interested in the extreme values of $\tilde{\pi}(v)$ ($0 \leq \pi_d \leq 1$), so we continue: let $x = \pi_d$,

$$\frac{d}{dx}(\frac{1-c+cx}{2-c+cx}) = \frac{c(2-c+cx) - c(1-c+cx)}{(2-c+cx)^2} = \frac{c}{(2-c+cx)^2} > 0.$$

So we find the maximum $\tilde{\pi}(v) = \frac{1}{2}$ for $\pi_d = 1$ and the minimum $\tilde{\pi}(v) = \frac{1-c}{2-c}$ for $\pi_d = 0$.

For equation 8 we have

$$\tilde{\pi}(v)(2-c) = 1 - c + c\tilde{\pi}_d$$

$$\tilde{\pi}(v) = \frac{1-c+c\tilde{\pi}_d}{2-c}$$

Again we are interested in the extreme values ($0 \leq \tilde{\pi}_d \leq 1 - \tilde{\pi}_v$). This is a linear equation so we find the maximum $\tilde{\pi}(v) = \frac{1}{2}$ for $\tilde{\pi}_d = 1 - \tilde{\pi}_v$ and the minimum $\tilde{\pi}(v) = \frac{1-c}{2-c}$ for $\tilde{\pi}_d = 0$.

Now we know

$$\frac{1-c}{2-c} \leq \tilde{\pi}(v) \leq \frac{1}{2}.$$

This implies

$$\frac{1}{2-c} \geq 1 - \tilde{\pi}(v) \geq \frac{1}{2}.$$

For $c = 0.85$ this gives

$$0.8696 \approx \frac{1}{2-0.85} \geq 1 - \tilde{\pi}(v) \geq \frac{1}{2}.$$

These provide boundaries for the precision of $\overline{\pi}$ calculated with the extra node solution.

Note that $\pi_d = 1$ implies that there are no non-dangling nodes in the original graph. In that case a random walk on the graph with the extra node consists of a jump from a dangling node to the extra node v with probability 1 followed by a jump from v to a dangling node with probability 1. It is clear that this will result in $\tilde{\pi}(v) = \frac{1}{2}$.

5 Algorithms to retrieve Points of View PageRank

In the previous section we discussed how to decompose the calculations to retrieve a Point of View PageRank. This decomposition is done because it reduces the amount of work to calculate a Point of View PageRank. The decomposition gives us parts that can be reused and combined during query time.

5.1 A remark on the pages in H

The idea behind the decomposition is to minimize the effect of the random walks not passing H. Therefore it is beneficial to choose H in such a way that the walks passing H have a relatively large probability. For a random starting point, these pages are precisely the high PageRank pages, because pages with high PageRank distribute more PageRank [3].

For the use of personalized PageRank however, the starting points are not random. Now it can be beneficial to choose H in another way. Further research is needed to find a H dependent on the starting points.

5.2 Double Taboo Algorithm

An instance of this algorithm in Matlab code:

```
% preH: adjacency matrix of the graph
% n: the size of the web
% H: substochastic matrix
% NOL: vector with ones for the dangling nodes, and zeros otherwise
[n H NOL] = zeroOneToSubStochastic(preH);

%G, the matrix for the extra node solution
%v, the distribution for the extra node
G = alpha * H;
v = (1/n)*ones(1,n);
v(1,n+1) = 0;
% create extra node
% link all nodes to this extra node
% with probability 1— alpha and the dangling nodes with
% probability 1
G(:,n+1) = (1—alpha)*ones(1,n) + alpha*NOL';
G(n+1,:) = v;
n = n+1;

% taboos: vector with indices of the taboo nodes as components
tabooSize = size(taboos,1);

% nottaboos: vector with indices of the non taboo nodes as components
nottaboos = setdiff(1:n,taboos);


% Gaa: rows and columns of G corresponding to the non taboo nodes
Gaa = G;
Gaa(taboos,:) = [];
Gaa(:,taboos) = [];

% Gab: rows of G corresponding to the non taboo nodes
% and columns of G corresponding to the taboo nodes
Gab = G;
Gab(taboos,:) = [];
Gab(:,nottaboos) = [];

% Gba: rows of G corresponding to the taboo nodes
% and columns of G corresponding to the non taboo nodes
Gba = G;
Gba(:,taboos) = [];
Gba(nottaboos,:) = [];

% Gbb: rows and columns of G corresponding to the taboo nodes
Gbb = G;
Gbb(nottaboos,:) = [];
Gbb(:,nottaboos) = [];

% pi,pi0: starting distribution
pi = 1/(n—tabooSize)*ones(1,n—tabooSize);
pi0 = pi;
```

```
GabtimesIminGbbInv = sparse(Gab*inv(speye(tabooSize)-Gbb));
M = GabtimesIminGbbInv*Gba;

prevpi = zeros(1,n-tabooSize);
residual = 1;
k = 1;

% Step 1 Aggregation
while (residual > eps)
    if(mod(k,2) == 1)
        pi = pi*M;
        residual=sum(abs(pi-prevpi));
        prevpi = pi;
    else
        residualb = 1;
        p = 1;
        pi0 = pi;

        % power algorithm
        while(residualb > eps)
            prevpib = pi;
            pi = pi * Gaa + pi0;
            p = p+1;
            display(p);
            residualb=sum(abs(pi-prevpib));
        end;
    end;
    k = k+1;
end;
sigmaa = pi';

% Step 2 Disaggregation
sigmab = sigmaa'* GabtimesIminGbbInv;

% building pi
pi(1:n-tabooSize) = sigmaa;
pi(n-tabooSize+1:n) = sigmab;
```

This algorithm is still in development.

6 Experiments

We did some experiments to test different kinds of algorithms. We use three different datasets.

6.1 Example graph

A graph of only five nodes. This graph was used to check whether an algorithm was correct.

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Node 2 and 4 are dangling nodes. For damping factor $c = 0.85$ the PageRank is

$$\overline{\pi} = \begin{bmatrix} 0.1822 \\ 0.1550 \\ 0.1550 \\ 0.3527 \\ 0.1550 \end{bmatrix}$$

6.2 Hollins dataset

A dataset of the hollins.edu domain crawled on January 15, 2004, which consists of 6012 nodes and is mentioned in [4]. It is now freely available on http://www.limfinity.com/ir/data/hollins.dat.gz. It has 23875 links, which makes the average number of incoming (or outgoing) links 3.97.

The maximum number of incoming links is 829 and the maximum number of outgoing links is 184. The number of dangling nodes is 3,189, which is 53% of the total number of nodes.

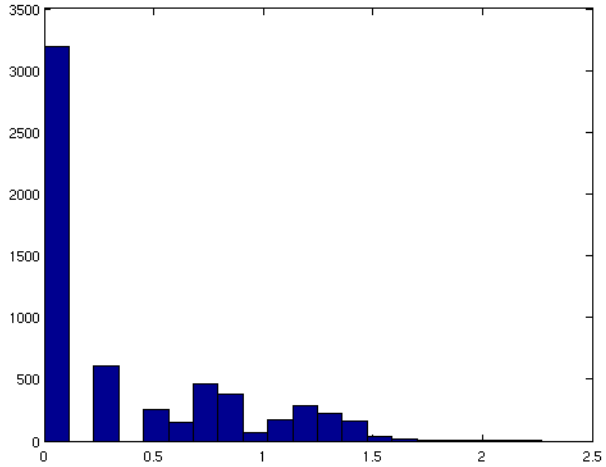

Figure 3: Outgoing links for Hollins dataset ($^{10}\log$)

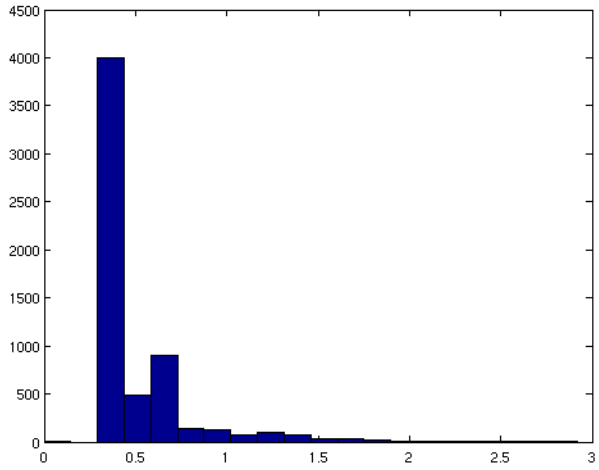

Figure 4: Incoming links for Hollins dataset ($^{10}\log$)

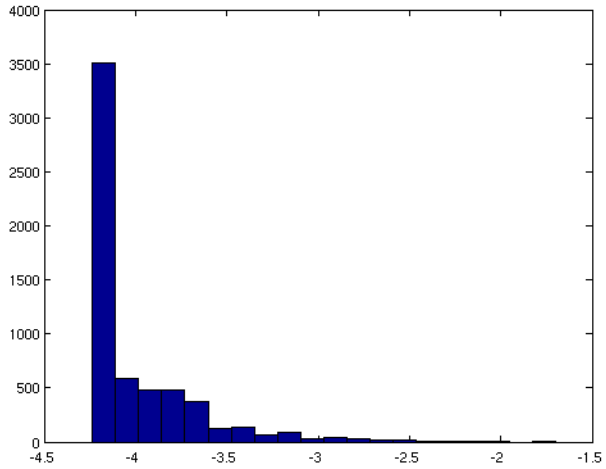

Figure 5: PageRank for Hollins dataset ($^{10}\log$) $c = 0.85$, 189 iterations

6.3 Stanford dataset

A dataset of the stanford.edu domain crawled in September 2002 by the Stanford WebBase project, which consists of 281,903 nodes and is and now freely available on http://kamvar.org/assets/data/stanford-web.tar.gz. It has 2,312,497 links which makes the average number of incoming (or outgoing) links 8.2. The maximum number of incoming links is 38,606 and the maximum number of outgoing links is 255. The number of dangling nodes is 172 (0.06%), which seems extremely low. For more information on possible drawbacks using this data set for testing purposes, see [19].

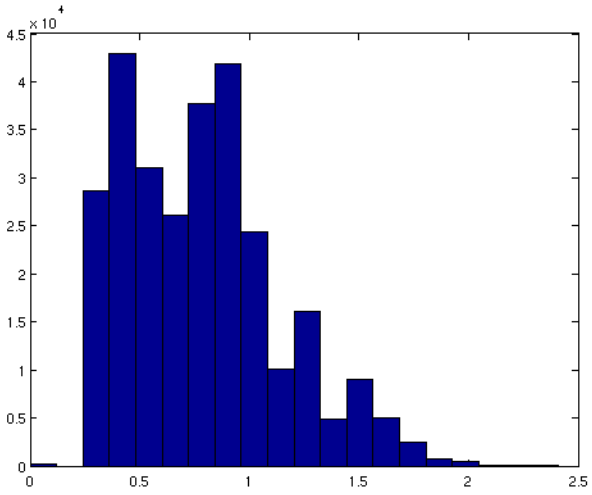

Figure 6: Outgoing links for Stanford dataset ($^{10}\log$)

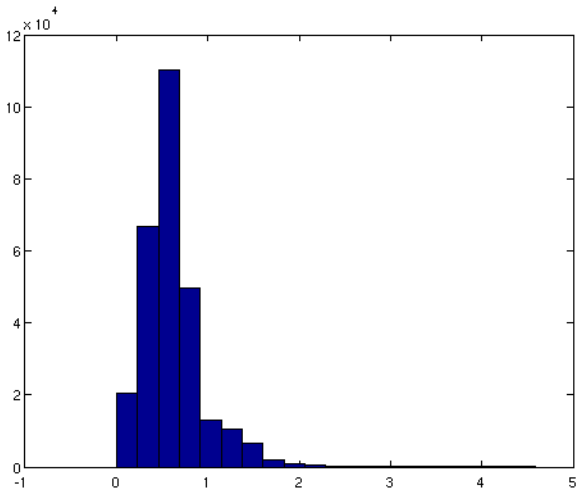

Figure 7: Incoming links for Stanford dataset ($^{10}\log$)

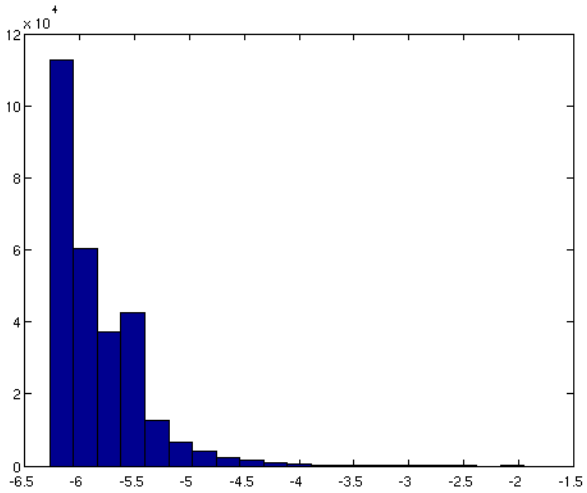

Figure 8: PageRank for Stanford dataset ($^{10}\log$) $c = 0.85$, 198 iterations

7 Discussion

7.1 Personalized search by Google

Google nowadays provides personalized results. As a user you can't have direct influence on how your results are personalized. The personalization is based on your Web History: a complete overview of all searches and sites visited by the (registered) user.

This restricts the way personalization is done. It is for example not possible to use the 'profile' of someone else to personalize your own results.

Although it is clear why Google provides personalization in this way[3], there may be valuable alternatives possible.

[3] Google has usability as a high value, so personalization should be as simple as possible.

7.2 Incoming links versus PageRank

A high number of incoming links has a big influence on the score of the PageRank. The x websites with the most incoming links were in our experiments the same as the x websites with the highest PageRank.

7.3 About "Scaling Personalized Search" [13]

The third algorithm called "Repeated Squaring Algorithm" is invalid.

The algorithm describes an iteration by doing

$$\boldsymbol{D_{2k}}[\boldsymbol{p}] = \boldsymbol{D_k}[\boldsymbol{p}] + \sum_{q \in Q_k(p)} E_k[p](q)\boldsymbol{D_k}[\boldsymbol{q}].$$

This may be a correct approach, but only $\boldsymbol{D_0}[\boldsymbol{p}] = \mathbf{0}$ is defined. In that case $\boldsymbol{D_{2k}}$ is not a very good approach, because of two reasons:

$$k = 0 \Rightarrow \boldsymbol{D_{2k}}[\boldsymbol{p}] = \boldsymbol{D_0}[\boldsymbol{p}] = \mathbf{0}$$

and

$$\boldsymbol{D_{2k}}[\boldsymbol{p}] = 0 + \sum_{q \in Q_k(p)} E_k[p](q)\mathbf{0} = \mathbf{0}.$$

In other words, this algorithm needs a specific $\boldsymbol{D_1}[\boldsymbol{p}]$, which - of course - may be achieved by one of the other two algorithms described in [13].

7.4 Using PageRank to select the webpages to crawl

On any moment in time outgoing links are known to exist to yet unknown pages. These pages can be handled as dangling nodes in the graph. Calculating PageRank can provide insight into which dangling nodes to check first. There is reason to assume that the dangling nodes with the highest PageRank have more value to the web than dangling nodes with lower PageRank.

References

[1] Lada A. Ademic. The small world web. *Proceedings of ECDL'99*, 1999.

[2] Jesse Alpert and Nissan Hajaj. We knew the web was big... *http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html*, July 2008.

[3] R. Anderson, C. Borgs, J. Chayes, J. Hopcraft, V. S. Mirrokni, and S.H. Teng. Local computation of pagerank contributions. *Algorithms and Models for the Web-Graph*, 2007.

[4] A.N.Langville and C.D. Meyer. *Google's PageRank and beyond*. Princeton University Press, 2006.

[5] K. Avrachenkov, N. Litvak, D. Nemirovsky, and N. Osipova. Monte carlo methods in pagerank computation: When one iteration is sufficient. *SIAM Journal on Numerical Analysis*, 2007.

[6] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30, 1998.

[7] K. L. Chung. *Markov chains with stationary transition probabilities*. Berlin: Springer-Verlag, 1967.

[8] M. Eirinaki and M. Vazirgiannis. Usage-based pagerank for web personalization. *Fifth IEEE International Conference on Data Mining*, 2005.

[9] T. Haveliwala. Topic-sensitive pagerank. *Proceedings of the Eleventh International World Wide Web Conference*, 2002.

[10] T. Haveliwala. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):784 – 796, July-Aug 2003.

[11] T. Haveliwala, S. Kamvar, and G. Jeh. An analytical comparison of approaches to personalizing pagerank. *Stanford University Technical Report*, 2003.

[12] I.C.F. Ipsen and T.M. Selee. Pagerank computation, with special attention to dangling nodes. *SIAM Journal on Matrix Analysis and Applications*, 29(4), 2007.

[13] G. Jeh and J. Widom. Scaling personalized web search. *International World Wide Web Conference*, 2003.

[14] S. Kamvar, T. Haveliwala, C. Manning, and G. Golub. Exploiting the block structure of the web for computing pagerank. *Stanford University Technical Report*, 2003.

[15] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 1999.

[16] Udi Manber. Introduction to google search quality. *http://googleblog.blogspot.com/2008/05/introduction-to-google-search-quality.html*, 2008.

[17] L. Page, S. Brin, R. Motwani, and T. Winograd. What can you do with a web in your pocket? *In Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 1998.

[18] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. *Stanford University Technical Report*, 1999.

[19] Sebastiano Vigna. Stanford matrix considered harmful. *http://drops.dagstuhl.de/opus/volltexte/2007/1058/pdf/07071.VignaSebastiano.Paper.1058.pdf*, 2007.