

**CS599: Structure and Dynamics of Networked Information (Spring 2005)**  
**01/19/2005: PageRank**  
**Scribes: Iftikhar Burhanuddin and Fragkiskos Papadopoulos**

In the previous lecture, we looked at and analyzed the HITS algorithm, based on Hubs and Authorities. This algorithm was designed with a scenario in mind where authorities might not cite each other (for example, Ford will be very unlikely to link to Chevrolet). Hence, HITS uses the idea of conferring authority indirectly through hubs.

In scenarios where authorities do cite each other, such as academic research work, it may be more appropriate to have direct conferral of authority between nodes. This is the idea behind the PageRank [2] algorithm.

Which algorithm is better may be hard to evaluate, and will depend on the query, among others. Among others, it raises the question of how one can even evaluate the performance of an algorithm when its objective function is not clearly defined.

## 1 PageRank

### 1.1 A first approach

Based on our intuition above, we would like to say that a page has high PageRank if it is pointed to by many pages of high PageRank. On the other hand, if a page points to many other pages, it presumably will not confer a lot of authority to all of them, but rather divide its authority evenly. This suggests the following update rule, starting from an arbitrary vector, such as  $P(i) = 1/n$  for all  $i$ : the new PageRank is  $P'(i) = \sum_{j \rightarrow i} \frac{1}{\delta^+(j)} \cdot P(j)$ , where  $\delta^+(j)$  is the out-degree of  $j$ .

We can immediately see that at any fixpoint of this update rule, the solution has to be the solution to a linear system in  $n$  variables. We could also converge to the fixpoint by iteratively applying the update rule. A more concise way of writing the update rule can be obtained as follows: If  $M$  is the square matrix with rows and columns corresponding to web pages, and  $M_{ij} = \frac{1}{\delta^+(i)}$  if  $i \rightarrow j$  and 0 otherwise, then, treating  $P(i)$  as a vector of web page ranks, the update rule can be written as  $P_{k+1}^{\vec{}} = M^T \cdot P_k^{\vec{}}$ . Notice that the operation has the useful property of “Mass Conservation”: the sum of the PageRanks of all pages is always exactly 1, as it is only redivided along outlinks.

### 1.2 A Problem and Solution

The above solution has a problem: it may not converge, and when it does, the result may not be what we are looking for. Specifically, if a web page has no outlinks (except, say, one to itself), then it will never pass on PageRank weight to any other nodes, but it will receive PageRank weight. Hence, all of the weight will collect at sinks, or, more generally, in the sink strongly connected components. All other nodes will have PageRank 0, which is certainly not corresponding to our intuition. In addition, which sinks will end up with all the weight will depend on the starting vector, so the PageRanks are not independent of the starting assignment, which is another undesirable property. Finally, in the case of a cycle of two nodes, the weight will oscillate between those nodes, but the weights will not converge.

There is a simple way to fix all of those problems at once. To motivate the approach, we notice that we can view  $M$  as the matrix of transition probabilities of a random walk on the web graph. When the random walk is at a node  $i$ , it chooses uniformly among all outgoing links, and follows that link. Hence, the first attempt above corresponds to computing the probabilities of being in a given state  $i$  in the limit of infinitely many steps. We can now modify the random walk as follows: for some small  $\epsilon \sim 1/7$ , with probability  $(1 - \epsilon)$ , the new Markov Chain does exactly the same as the old random walk. With the remaining probability  $\epsilon$ , the new Markov Chain chooses a uniformly random vertex and jumps to it. This random process can intuitively

be motivated by the model of a random surfer who gets bored and jumps to a uniformly random page with probability  $1/7$  in each step.

In terms of matrices, we can express the new process as follows. Let  $1_{p \times q}$  denote the  $p \times q$  matrix of all ones. The update step from the distribution  $P_k$  (at some time  $k$ ) to  $P_{k+1}$  can now be written as follows:

$$\begin{aligned} \vec{P}_{k+1} &= (1 - \epsilon)M^T \cdot \vec{P}_k + \epsilon \cdot \frac{1}{n} 1_{n \times 1} \\ &= ((1 - \epsilon)M^T + \epsilon \cdot \frac{1}{n} 1_{n \times n}) \cdot \vec{P}_k \\ &=: M' \cdot \vec{P}_k \end{aligned}$$

To show that this new version of PageRank converges, we can use the following well-known theorem about Markov Chains.

**Theorem 1** *If the Markov Chain  $M$  is irreducible and aperiodic (the gcd of all cycle lengths is 1), then it converges to a unique stationary probability  $\vec{\pi}$ , i.e.,  $\vec{\pi} = M^T \cdot \vec{\pi}$ .*

The term “irreducible” means that the corresponding graph (of non-zero entries of  $M$ ) is strongly connected. Certainly, our matrix  $M'$  is both irreducible and aperiodic, as we added a jump probability of  $\epsilon/n$  between any pair of vertices, i.e., the graph is now complete. (Notice that for  $M$  that are not aperiodic, oscillations might occur, as in the example mentioned above of a cycle of length 2. Similarly, the probabilities of the random walk may depend on where the random walk started. This also applies if  $M$  is not irreducible. On the other hand, if  $G$  is irreducible and aperiodic, then the probabilities are independent of the starting point.)

For students who have seen this before, a very simple Markov Chain coupling argument (about the simplest possible) shows that the mixing time of this chain is  $1/\epsilon$ , i.e., a constant. As a result, the convergence is exponentially fast: within  $O(\log \frac{1}{\delta})$  iterations, the error is at most  $\delta$ .

Regarding computation, we are now dealing with a very large and dense matrix. However, the fact that most entries are equal to  $\epsilon/n$  allows for efficient matrix computation techniques to be applied nevertheless.

The original idea of PageRank is that the PageRank  $\pi_i$  of a page  $i$  is an “intrinsic” measure of its quality. At query time, we merely need to find all text-based matches and their neighborhood, and output them sorted by PageRank.

If we do the processing at query time instead, we might be able to obtain better results by using text-based relevance. For instance, we can prune the node set by starting from some 200 best text-based matches, and creating a bag of words from them. Then, pages with very different bags of words can be pruned out, i.e., if the inner product  $\vec{b} \cdot \vec{w}$  between the occurrence vectors  $\vec{w}$  and  $\vec{b}$  is small. In addition, we can gauge the importance of links by looking for relevant text near the anchor. A relevant empirical observation is that most links have relevant text between 25 bytes before and 50 bytes after their own occurrence [3].

## 2 Topic-sensitive PageRank

In reality, we can neither compute all PageRanks at query time, nor can we pre-compute the PageRanks for all possible queries. A middle path has been proposed by Haveliwala [4]. The idea is to precompute PageRanks for a few landmark topics, and then express queries as a combination of these topics, and calculate the PageRanks from the pre-computed ones.

Assume that there are  $T$  topics, and topic  $t$  is characterized by its restart probability vector  $\vec{f}_t$ . The corresponding update rule is thus  $\vec{P}' = (1 - \epsilon)M^T \cdot \vec{P} + \epsilon \cdot \vec{f}_t$ . By defining the matrix  $F_t = [\vec{f}_t \vec{f}_t \dots \vec{f}_t]$ , we can express this new rule as

$$\begin{aligned} \vec{P}' &= ((1 - \epsilon)M^T + \epsilon \cdot F_t) \cdot \vec{P} \\ &=: M_t \cdot \vec{P} \end{aligned}$$

The corresponding PageRanks for topics  $t$  are now the stationary probabilities  $\pi_{\vec{f}_t}$  of  $M_t$ .

From the PageRanks for topics  $t$ , we can compute those for queries  $q$  as follows. We can consider a query as a convex combination of topics, i.e.  $q = \sum_t \gamma_t \cdot t$  (where  $t$  just denotes a topic, and  $\sum_t \gamma_t = 1$ ). Then, we can identify with  $q$  the reset vector  $\vec{f}_q = \sum_t \gamma_t \vec{f}_t$ . The interesting observation is that the corresponding stationary probabilities  $\pi_{\vec{f}_q}$  can be obtained as convex combinations of the probabilities for the landmark topics:

**Lemma 2**  $\pi_{\vec{f}_q} = \sum_t \gamma_t \cdot \pi_{\vec{f}_t}$

**Proof.** We will show that the vector on the right is stationary for  $M_q$ . As the stationary distribution is unique, this proves that it is equal to  $\pi_{\vec{f}_q}$ . In order to do so, we use the fact that each  $\pi_{\vec{f}_t}$  is stationary for its corresponding  $M_t$ , and then use the linearity of matrix-vector multiplication and summation:

$$\begin{aligned} \sum_t \gamma_t \cdot \pi_{\vec{f}_t} &= \sum_t \gamma_t \cdot ((1 - \epsilon)M^T \cdot \pi_{\vec{f}_t} + \epsilon \cdot \vec{f}_t) \\ &= (1 - \epsilon)M^T \sum_t \gamma_t \cdot \pi_{\vec{f}_t} + \epsilon \sum_t \gamma_t \vec{f}_t \\ &= (1 - \epsilon)M^T (\sum_t \gamma_t \cdot \pi_{\vec{f}_t}) + \epsilon \vec{f}_q \\ &= M_q \cdot (\sum_t \gamma_t \cdot \pi_{\vec{f}_t}) \end{aligned}$$

■

Hence, the PageRanks of linear combinations of topics can be efficiently computed from precomputed topic PageRanks.

### 3 Single-Word Queries

To go even further, one could try to pre-compute the PageRanks for every single-word query. At first, this may seem very daunting, as the number of words is far in excess of 100000, and hence, it appears as though the storage requirement would be larger than  $10^5 \cdot 10^9 = 10^{14}$  PageRank values. However, more careful indexing may reduce this requirement significantly, as most words do not appear in most pages. A simple back-of-the-envelope calculation observed by Domingos and Richardson [5], goes as follows. Let  $w$  denote a word, and  $i$  a page. Further, let  $p_w$  be the number of pages containing  $w$ , and  $s_i$  the number of words contained in page  $i$ , and  $x_{w,i} = 1$  if word  $w$  appears in page  $i$ . Then, the total required index size is

$$\sum_w p_w = \sum_{w,i} x_{w,i} = \sum_i s_i.$$

Notice that the average page contains only about a few hundred words, so the last sum is only about a few hundred times the number  $n$  of pages in the web. While this is not yet quite feasible, it is not too far removed from current technology.

For multi-word queries, the situation is a lot more bleak: the number of distinct pairs that appear in web pages is much higher, and the same kind of simplistic analysis does not work: the sum now becomes  $\sum_i s_i^2$ , which may be much larger. So far, no good ideas seem to be around about whether or how one could perform useful precomputations for multi-word queries.

## References

- [1] Kleinberg, J. *Authoritative sources in a hyperlinked environment*. Proc. 9th ACM-SIAM Symposium on Discrete Algorithms, 1998.
- [2] Brin, S. and Page, L. *The Anatomy of a Large-Scale Hypertextual Web Search Engine*. Proc. 7th International World Wide Web Conference, 1998.
- [3] Chakrabarti, S. *Mining the link structure of the World Wide Web*. IEEE Computer, August 1999.
- [4] Haveliwala, T. *Topic-Sensitive PageRank*. Proc. 11th International World Wide Web Conference, May 2002.
- [5] Domingos, P. and Richardson, M. *The Intelligent Surfer: Probabilistic Combination of Link and Content Information in PageRank*. Advances in Neural Information Processing Systems 14, 2002.