Efficient PageRank Approximation via Graph Aggregation

Andrei Z. Broder*
IBM T.J. Watson Research Center
Hawthorne, NY, USA
abroder@us.ibm.com

Ronny Lempel*
IBM Haifa Research Labs, Haifa, Israel
rlempel@il.ibm.com

Farzin Maghoul
Yahoo! Inc.
farzin.maghoul@overture.com

Jan Pedersen
Yahoo! Inc.
jan.pedersen@overture.com

ABSTRACT

We present a framework for approximating random-walk based probability distributions over Web pages using graph aggregation. We (1) partition the Web's graph into *classes* of quasi-equivalent vertices, (2) project the page-based random walk to be approximated onto those classes, and (3) compute the stationary probability distribution of the resulting class-based random walk. From this distribution we can quickly reconstruct a distribution on pages. In particular, our framework can approximate the well-known PageRank distribution by setting the classes according to the set of pages on each Web host. We experimented on a Web-graph containing over 1.4 billion pages, and were able to produce a ranking that has Spearman rank-order correlation of 0.95 with respect to PageRank. A simplistic implementation of our method required less than half the running time of a highly optimized implementation of PageRank, implying that larger speedup factors are probably possible.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Performance, Experimentation, Measurements

Keywords

Link analysis, search engines, Web information retrieval

1. INTRODUCTION

Since the late nineties, Web search engines have started to rely more and more on off-page, Web-specific data such as link analysis, anchor-text, and click-through data. One particular form of link-based ranking factors are static scores, which are query-independent importance scores that are assigned to all Web pages. The most famous algorithm for producing such scores is *PageRank*, devised by Brin and Page while developing the ranking module for the prototype of the search engine *Google* (`www.google.com`)[1]. PageRank can be described as the stationary probability distribution of a certain random walk on the Web graph - the graph whose nodes are the Web pages, and whose directed edges are the hyperlinks between pages. However, modern search engines index billions of pages, interconnected by tens of billions of links. Computing PageRank on this scale requires considerable resources, both in terms of CPU cycles and in terms of random-access memory. Clock-wall times for PageRank computations on large graphs can reach many hours. Furthermore, topic-induced and personalized PageRank flavors [3, 4] require PageRank computations to be performed multiple times. Thus, speedy implementations of PageRank become critical. Indeed, many research efforts have investigated accelerations of PageRank-like computations, e.g. [2, 5].

To some extent, our work follows in this vein. We introduce a framework for computing PageRank-like probability distributions for Web pages based on graph aggregation. The basic idea is to partition the graph into *classes* of quasi-equivalent vertices, and to compute the stationary probability distribution of a biased random walk on classes. From this distribution we can reconstruct a distribution on pages. In the context of this paper, the classes used were the sets of pages on a given host. Our host-aggregated random walk is not equivalent to PageRank. Rather, it closely approximates PageRank. On a Web-graph containing over 1.4 billion pages and 6.6 billion links, it produced a ranking that has Spearman rank-order correlation of 0.95 with respect to PageRank. Furthermore, a simplistic implementation of our method required less than half the running time of a highly optimized implementation of PageRank.

2. PAGERANK

PageRank quantifies the importance of Web pages. The PageRank of a page p is the probability of visiting p in a random walk of the entire Web, where the set of states of the random walk is the set of Web pages, and each random step is of one of the following two types: (1) Choose a Web page uniformly at random, and jump to it, or (2) From the given state/page q, choose at random an outgoing link of q, and follow that link to the destination page [1].

PageRank chooses a parameter $d,\ 0 < d < 1$; each state transition is of the first transition type with probability $1 - d$, and of the second type with probability d. Naturally, this random walk can be represented by a stochastic matrix, whose principal eigenvector corresponds to the stationary distribution of the walk. Thus, PageRank scores are typically computed by applying the Power method for eigenvector approximation, which involves repeated multiplications of an arbitrary initial vector by the matrix in question, until the iterations converge to a fixed vector.

*Significant portions of the work presented here were done while these authors were employed by the AltaVista corporation.

WWW2004, May 17–22, 2004, New York, New York, USA.
ACM 1-58113-912-8/04/0005.

[1]Pages that have no outgoing links are treated as if they link to all other Web pages in many PageRank-related papers.

3. THE GRAPH AGGREGATION METHOD

Let T be a random walk on a graph with n nodes. T will denote both the random walk and the stochastic matrix which governs it. Let the n nodes be partitioned into m *classes* $H_1, \ldots, H_m$. We concentrate on the case where T denotes PageRank and the partitioning of Web pages is according to their host. Hence, a class H_i contains all the nodes (pages) on a certain host. We develop an alternative random walk T', derived from T, whose stationary distribution can be computed more efficiently. In T', a state transition departing from a node $x \in H_i$ is the following two-stage process:

- Move to some node $y \in H_i$ according to a distribution π_{H_i}. Note that π_{H_i} depends only on the class of x.
- From y, perform a transition according to T.

The algorithm for calculating the stationary distribution of T':

1. Define an $m \times m$ stochastic matrix $\tilde{T} = [\tilde{t}_{i,j}]$ as follows:
$$\tilde{t}_{H_i,H_j} = \sum_{q \in H_i} \pi_{H_i}(q) \cdot \sum_{p \in H_j} t_{q,p} \; .$$
2. Calculate the principal eigenvector of $\tilde{T}$: Compute an m-dimensional probability distribution $\tilde{\alpha}$ satisfying $\tilde{\alpha}\tilde{T} = \tilde{\alpha}$.
3. Compute an n-dimensional probability distribution γ, where for each node p, $\gamma(p) = \tilde{\alpha}_{h(p)} \cdot \pi_{h(p)}(p)$ ($h(p)$ denotes the class to which p belongs).
4. The stationary distribution of T' is the vector $\beta \stackrel{\triangle}{=} \gamma T$.

Computational demands: the standard power-iteration computation of PageRank converges in a few dozen iterations. Each iteration requires one pass over the complete list of links for the entire Web graph. In contrast, our algorithm needs only two passes over the entire set of links (steps $1, 4$). Each power iteration required by our algorithm (in step 2) is linear in the number of links of the host-graph, which is typically much smaller (maybe by a factor of 20), than the number of links in the page-graph. This has implications also on the memory demands of our algorithm: search engines compute connectivity-based measures over tens of billions of links. This scale of data exceeds the RAM capabilities of most single-machine platforms. However, our algorithm performs most of its computations on the much smaller host graph, which may fit in the RAM of a single machine.

This algorithm is related to the *BlockRank* algorithm [5]. BlockRank accelerates PageRank by deriving a distribution vector v, from which (empirically) fewer Power iterations are required until convergence. The vector v is closely related to the vector γ produced by step 3 above, when each distribution π_H is chosen as the intra-host PageRank vector of H. Since BlockRank multiplies v by T several times (until convergence) while we only multiply γ by T once (step 4), our approach is speedier than BlockRank.

Experiments: we conducted experiments on a Web-graph containing over 1446 million pages with almost 6650 million links, from a crawl of the Web conducted by AltaVista in September 2003. The number of unique hosts in this graph was about 31 million, with 241 million host-to-host edges. We set all intra-host distributions π_H to be *uniform*; hence, we called the resulting random walk flavor the "U-model". Computing PageRank on an Alpha server with four 667 MHz CPUs required 12.5 hours, while computing the U-model scores took 5.8 hours (a speedup factor of about 2.1). The PageRank computations used a robust and optimized infrastructure, whereas our modified algorithm was written in an ad-hoc, non-optimized manner. We predict that by optimizing our implementation, speedup factors can approach the full potential indicated by the ratio of the number links in the Web graph and the corresponding figure in the host-graph.

We measured the correlation between PageRank and the U-model using a sample of 1298 pages. The sample was not a uniform random sample since, by virtue of the power-law distribution of PageRank, the bulk of Web pages have few or no inlinks. Such pages would be ranked similarly by practically any static score algorithm. Instead, we sorted pages according to their PageRank, and sampled as follows: each of the top 1000 pages was chosen with probability 0.2. Then, for $j = 3, \ldots, 8$, the pages in places $1 + 10^j, \ldots, 10^{j+1}$ were sampled w.p. $0.2 \cdot 10^{2-j}$. Thus, the sample covered the full range of PageRank values, and included many more pages with high PageRank values than a uniform random sample would have produced. In this sample, U-model had Pearson correlation 0.81 with PageRank. Furthermore, we compared the *rankings* that are induced by the two score flavors. The Spearman rank-order correlation between U-model and PageRank was quite high, at 0.95. This suggests that U-model can be used as an effective approximation of PageRank in relevance ranking. Whether the (small) differences between U-model and PageRank would imply better or worse search quality is currently unknown.

4. CONCLUSIONS

This paper introduced a new framework for calculating random-walk based static ranks of pages on the Web. The framework approximates the behavior of a given random walk on the Web's graph in a scalable manner, by performing most of its calculations on a more compact representation of the graph - a representation that follows from aggregating multiple pages onto a single node.

We experimented with a model of random Web browsing that decouples intra-host and inter-host steps. Departing from a page in our model consists of a random transition to another page of the same host, followed by a PageRank-like step from that page. Whereas PageRank requires repeated scans of the Web-graph's links, most of the computations required by our approach involve scanning the edges of the Web's host-graph.

Future research should experiment with different aggregates of the Web's graph, and with different stochastic behaviors within those aggregates. Each such flavor should be evaluated in terms of the speedup factors it achieves, and in terms of the search quality that it induces when used in the ranking core of search engines.

5. REFERENCES

[1] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proc. 7th International WWW Conference*, pages 107–117, 1998.

[2] T. H. Haveliwala. Efficient computation of pagerank. Technical Report Technical Report, Stanford University, October 1999.

[3] T. H. Haveliwala. Topic-sensitive pagerank. In *Proc. 11th International WWW Conference (WWW2002)*, 2002.

[4] G. Jeh and J. Widom. Scaling personalized web search. In *Proc. 12th International WWW Conference (WWW2003), Budapest, Hungary*, pages 271–279, 2003.

[5] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub. Exploiting the block structure of the web for computating pagerank. Technical Report Technical Report, Stanford University, March 2003.