Computing PageRank using Power Extrapolation

Taher Haveliwala, Sepandar Kamvar, Dan Klein, Chris Manning, and Gene Golub

Stanford University

Abstract. We present a novel technique for speeding up the computation of PageRank, a hyperlink-based estimate of the "importance" of Web pages, based on the ideas presented in [7]. The original PageRank algorithm uses the Power Method to compute successive iterates that converge to the principal eigenvector of the Markov matrix representing the Web link graph. The algorithm presented here, called Power Extrapolation, accelerates the convergence of the Power Method by subtracting off the error along several nonprincipal eigenvectors from the current iterate of the Power Method, making use of known nonprincipal eigenvalues of the Web hyperlink matrix. Empirically, we show that using Power Extrapolation speeds up PageRank computation by 30% on a Web graph of 80 million nodes in realistic scenarios over the standard power method, in a way that is simple to understand and implement.

1 Introduction

The PageRank algorithm for determining the "importance" of Web pages has become a central technique in Web search [9]. The core of the PageRank algorithm involves computing the principal eigenvector of the Markov matrix representing the hyperlink structure of the Web. As the Web graph is very large, containing over a billion nodes, the PageRank vector is generally computed offline, during the preprocessing of the Web crawl, before any queries have been issued.

The development of techniques for computing PageRank efficiently for Web-scale graphs is important for a number of reasons. For Web graphs containing a billion nodes, computing a PageRank vector can take several days. Computing PageRank quickly is necessary to reduce the lag time from when a new crawl is completed to when that crawl can be made available for searching. Furthermore, recent approaches to personalized and topic-sensitive PageRank schemes [2, 10, 6] require computing *many* PageRank vectors, each biased towards certain types of pages. These approaches intensify the need for faster methods for computing PageRank.

A practical extrapolation method for accelerating the computation of PageRank was first presented by Kamvar et al. in [7]. That work assumed that none of the nonprincipal eigenvalues of the hyperlink matrix were known. Haveliwala and Kavmar derived the modulus of the second eigenvalue of the hyperlink matrix in [3]. By exploiting known eigenvalues of the hyperlink matrix, we derive here a simpler and more effective extrapolation algorithm. We show empirically on an 80 million page Web crawl that this algorithm speeds up the computation of PageRank by 30% over the standard power method. The speedup in number of iterations is basically equivalent to that of the Quadratic Extrapolation algorithm introduced in [7], but the method presented here is much simpler to implement, and has negligible overhead, so that its wallclock-speedup is higher by 9%.[1]

2 Preliminaries

In this section we summarize the definition of PageRank [9] and review some of the mathematical tools we will use in analyzing and improving the standard iterative algorithm for computing PageRank.

Underlying the definition of PageRank is the following basic assumption. A link from a page $u \in Web$ to a page $v \in Web$ can be viewed as evidence that v is an "important" page. In particular, the amount of importance conferred on v by u is proportional to the importance of u and inversely proportional to the number of pages u points to. Since the importance of u is itself not known, determining the importance for every page $i \in Web$ requires an iterative fixed-point computation.

To allow for a more rigorous analysis of the necessary computation, we next describe an equivalent formulation in terms of a random walk on the directed Web graph G. Let $u \rightarrow v$ denote the existence of an edge from u to v in G. Let $\deg(u)$ be the outdegree of page u in G. Consider a random surfer visiting page u at time k. In the next time step, the surfer chooses a node v_i from among u's out-neighbors $\{v|u \rightarrow v\}$ uniformly at random. In other words, at time $k+1$, the surfer lands at node $v_i \in \{v|u \rightarrow v\}$ with probability $1/\deg(u)$.

The PageRank of a page i is defined as the probability that at some particular time step $k > K$, the surfer is at page i. For sufficiently large K, and with minor modifications to the random walk, this probability is unique, illustrated as follows. Consider the Markov chain induced by the random walk on G, where the states are given by the nodes in G, and the stochastic transition matrix describing the transition from i to j is given by P with $P_{ij} = 1/\deg(i)$.

For P to be a valid transition probability matrix, every node must have at least 1 outgoing transition; i.e., P should have no rows consisting of all zeros. This holds if G does not have any pages with outdegree 0, which does not hold for the Web graph. P can be converted into a valid transition matrix by adding a complete set of outgoing transitions to pages with outdegree 0. In other words, we can define the new matrix P' where all states have at least one outgoing transition in the following way. Let n be the number of nodes (pages) in the Web graph. Let $\boldsymbol{v}$ be the n-dimensional column vector representing a uniform probability distribution over all nodes:

$$\boldsymbol{v} = [\frac{1}{n}]_{n \times 1} \tag{1}$$

Let $\boldsymbol{d}$ be the n-dimensional column vector identifying the nodes with outdegree 0:

$$d_i = \begin{cases} 1 & \text{if } \deg(i) = 0, \\ 0 & \text{otherwise} \end{cases}$$

[1] Note that as the Quadratic Extrapolation algorithm of [7] does not assume the second eigenvalue of the matrix is known, it is more widely applicable (but less efficient) than the algorithms presented here.

```
$\boldsymbol{y} = cP^T\boldsymbol{x}$;
$w = ||\boldsymbol{x}||_1 - ||\boldsymbol{y}||_1$;
$\boldsymbol{y} = \boldsymbol{y} + w\boldsymbol{v}$;
```

Algorithm 1: Computing $\boldsymbol{y} = A\boldsymbol{x}$

Then we construct P' as follows:

$$D = \boldsymbol{d} \cdot \boldsymbol{v}^T$$
$$P' = P + D$$

In terms of the random walk, the effect of D is to modify the transition probabilities so that a surfer visiting a dangling page (i.e., a page with no outlinks) randomly jumps to another page in the next time step, using the distribution given by $\boldsymbol{v}$.

By the Ergodic Theorem for Markov chains [1], the Markov chain defined by P' has a unique stationary probability distribution if P' is aperiodic and irreducible; the former holds for the Markov chain induced by the Web graph. The latter holds iff G is strongly connected, which is generally *not* the case for the Web graph. In the context of computing PageRank, the standard way of ensuring this property is to add a new set of complete outgoing transitions, with small transition probabilities, to *all* nodes, creating a complete (and thus strongly connected) transition graph. In matrix notation, we construct the irreducible Markov matrix P'' as follows:

$$E = [1]_{n\times 1} \times \boldsymbol{v}^T$$
$$P'' = cP' + (1-c)E$$

In terms of the random walk, the effect of E is as follows. At each time step, with probability $(1-c)$, a surfer visiting any node will jump to a random Web page (rather than following an outlink). The destination of the random jump is chosen according to the probability distribution given in $\boldsymbol{v}$. Artificial jumps taken because of E are referred to as *teleportation*.

By redefining the vector $\boldsymbol{v}$ given in Equation 1 to be nonuniform, so that D and E add artificial transitions with nonuniform probabilities, the resultant PageRank vector can be biased to prefer certain kinds of pages. For this reason, we refer to $\boldsymbol{v}$ as the *personalization* vector.

For simplicity and consistency with prior work, the remainder of the discussion will be in terms of the transpose matrix, $A = (P'')^T$; i.e., the transition probability distribution for a surfer at node i is given by row i of P'', and column i of A.

Note that the edges artificially introduced by D and E never need to be explicitly materialized, so this construction has no impact on efficiency or the sparsity of the matrices used in the computations. In particular, the matrix-vector multiplication $\boldsymbol{y} = A\boldsymbol{x}$ can be implemented efficiently using Algorithm 1.

Assuming that the probability distribution over the surfer's location at time 0 is given by $\boldsymbol{x}^{(0)}$, the probability distribution for the surfer's location at time k is given by $\boldsymbol{x}^{(k)} = A^k\boldsymbol{x}^{(0)}$. The unique stationary distribution of the Markov chain is defined as

$\lim_{k\to\infty} x^{(k)}$, which is equivalent to $\lim_{k\to\infty} A^k x^{(0)}$, and is independent of the initial distribution $\boldsymbol{x}^{(0)}$. This is simply the principal eigenvector of the matrix $A = (P'')^T$, which is exactly the PageRank vector we would like to compute.

The standard PageRank algorithm computes the principal eigenvector by starting with the uniform distribution $\boldsymbol{x}^{(0)} = \boldsymbol{v}$ and computing successive iterates $\boldsymbol{x}^{(k)} = A\boldsymbol{x}^{(k-1)}$ until convergence. This is known as the Power Method, and is discussed in further detail in Section 4.

We present here a technique, called Power Extrapolation, that accelerates the convergence of the Power Method by subtracting off the first few nonprincipal eigenvectors from the current iterate $\boldsymbol{x}^{(k)}$. We take advantage of the fact that the first eigenvalue of A is 1 (because A is stochastic), and the modulus of the second eigenvalues is c (see [3]), to compute estimates of the error along nonprincipal eigenvectors using two iterates of the Power Method. This Power Extrapolation calculation is easy to integrate into the standard PageRank algorithm and yet provides substantial speedups.

3 Experimental Setup

In the following sections, we will be introducing a series of algorithms for computing PageRank, and discussing the rate of convergence achieved on realistic datasets. Our experimental setup was as follows. We used the LARGEWEB link graph, generated from a crawl of the Web by the Stanford WebBase project in January 2001 [4]. LARGEWEB contains roughly 80M nodes, with close to a billion links, and requires 3.6GB of storage. Dangling nodes were removed as described in [9]. The graph was stored using an adjacency list representation, with pages represented by 4-byte integer identifiers. On an AMD Athlon 1533MHz machine with a 2-way linear RAID disk volume and 3.5GB of main memory, each iteration of Algorithm 1 on the 80M page LARGEWEB dataset takes roughly 10 minutes. Given that the full Web contains billions of pages, and computing PageRank generally requires roughly 50 applications of Algorithm 1, the need for fast methods is clear.

We measured the relative rates of convergence of the algorithms that follow using the L_1 norm of the residual vector; i.e.,

$$||Ax^{(k)} - x^{(k)}||_1$$

See [7] for discussion of measures of convergence for page importance algorithms.

4 Power Method

4.1 Formulation

One way to compute the stationary distribution of a Markov chain is by explicitly computing the distribution at successive time steps, using $\boldsymbol{x}^{(k)} = A\boldsymbol{x}^{(k-1)}$, until the distribution converges.

This leads us to Algorithm 2, the Power Method for computing the principal eigenvector of A. The Power Method is the oldest method for computing the principal eigenvector of a matrix, and is at the heart of both the motivation and implementation of the original PageRank algorithm (in conjunction with Algorithm 1).

```
function $\boldsymbol{x}^{(n)} = PowerMethod()$ {
$\boldsymbol{x}^{(0)} = \boldsymbol{v}$;
$k = 1$;
repeat
    $\boldsymbol{x}^{(k)} = A\boldsymbol{x}^{(k-1)}$;
    $\delta = ||x^{(k)} - x^{(k-1)}||_1$;
    $k = k + 1$;
until $\delta < \epsilon$;
}
```

Algorithm 2: Power Method

The intuition behind the convergence of the power method is as follows. For simplicity, assume that the start vector $\boldsymbol{x}^{(0)}$ lies in the subspace spanned by the eigenvectors of A.[2] Then $\boldsymbol{x}^{(0)}$ can be written as a linear combination of the eigenvectors of A:

$$\boldsymbol{x}^{(0)} = \boldsymbol{u}_1 + \alpha_2\boldsymbol{u}_2 + \ldots + \alpha_m\boldsymbol{u}_m \tag{2}$$

Since we know that the first eigenvalue of a Markov matrix is $\lambda_1 = 1$,

$$\boldsymbol{x}^{(1)} = A\boldsymbol{x}^{(0)} = \boldsymbol{u}_1 + \alpha_2\lambda_2\boldsymbol{u}_2 + \ldots + \alpha_m\lambda_m\boldsymbol{u}_m \tag{3}$$

and

$$\boldsymbol{x}^{(n)} = A^n\boldsymbol{x}^{(0)} = \boldsymbol{u}_1 + \alpha_2\lambda_2^n\boldsymbol{u}_2 + \ldots + \alpha_m\lambda_m^n\boldsymbol{u}_m \tag{4}$$

Since $\lambda_n \leq \ldots \leq \lambda_2 < 1$, $A^{(n)}\boldsymbol{x}^{(0)}$ approaches $\boldsymbol{u}_1$ as n grows large. Therefore, the Power Method converges to the principal eigenvector of the Markov matrix A.

4.2 Operation Count

A single iteration of the Power Method consists of the single matrix-vector multiply $A\boldsymbol{x}^{(k)}$. Generally, this is an $O(n^2)$ operation. However, if the matrix-vector multiply is performed as in Algorithm 1, the matrix A is so sparse that the matrix-vector multiply is essentially $O(n)$. In particular, the average outdegree of pages on the Web has been found to be around 7 [8]. On our datasets, we observed an average of around 8 outlinks per page.

It should be noted that if λ_2 is close to 1, then the power method is slow to converge, because n must be large before λ_2^n is close to 0.

4.3 Results and Discussion

As we show in [3], the eigengap $1 - |\lambda_2|$ for the Web Markov matrix A is given exactly by the teleport probability $1 - c$. Thus, when the teleport probability is large, the Power Method works reasonably well. However, for a large teleport probability (and with a

[2] This assumption does not affect convergence guarantees.

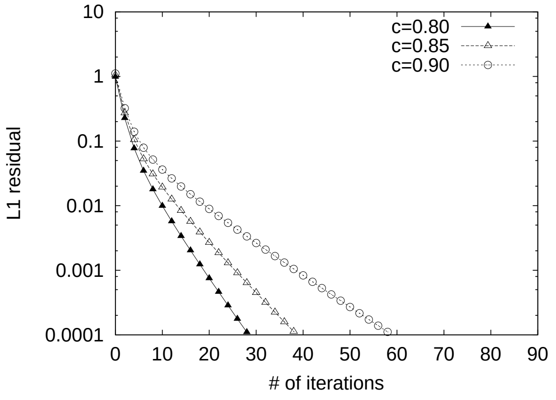

Fig. 1. Comparison of convergence rate for the standard Power Method on the LargeWeb dataset for $c \in \{0.80, 0.85, 0.90\}$.

uniform personalization vector $\boldsymbol{v}$), the effect of link spam is increased, and pages can achieve unfairly high rankings.[3]

In Figure 1, we show the convergence on the LargeWeb dataset of the Power Method for $c \in \{0.80, 0.85, 0.90\}$ using a uniform damping vector $\boldsymbol{v}$. Note that increasing c slows down convergence. Since each iteration of the Power Method takes 10 minutes, computing 50 iterations requires over 8 hours. As the full Web is estimated to contain over two billion static pages, using the Power Method on Web graphs close to the size of the Web would require several days of computation.

In the next sections, we describe how to remove the error components of $x^{(k)}$ along the directions of the nonprincipal eigenvectors, thus increasing the effectiveness of Power Method iterations.

5 Extrapolation Methods

A practical extrapolation method for accelerating the computation of PageRank was first presented by Kamvar et al. in [7]. That work assumed that none of the nonprincipal eigenvalues of the hyperlink matrix are known. However, Haveliwala and Kamvar [3] proved that the modulus of the second eigenvalue of A is given by the damping factor c. Note that the web graph can have many eigenvalues with modulus c (i.e., one of c, $-c$, ci, and $-ci$). In this section, we present a series of algorithms that exploit known eigenvalues of A to accelerate the Power Method for computing PageRank.

[3] A high teleport probability means that every page is given a fixed “bonus” rank. Link spammers can make use of this bonus to generate local structures to inflate the importance of certain pages.

5.1 Simple Extrapolation

Formulation The simplest extrapolation rule assumes that the iterate $\boldsymbol{x}^{(k-1)}$ can be expressed as a linear combination of the eigenvectors u_1 and u_2, where u_2 has eigenvalue c.

$$\boldsymbol{x}^{(k-1)} = \boldsymbol{u}_1 + \alpha_2 \boldsymbol{u}_2 \tag{5}$$

Now consider the current iterate $\boldsymbol{x}^{(k)}$; because the Power Method generates iterates by successive multiplication by A, we can write $\boldsymbol{x}^{(k)}$ as

$$\begin{aligned} \boldsymbol{x}^{(k)} &= A\boldsymbol{x}^{(k-1)} && (6) \\ &= A(\boldsymbol{u}_1 + \alpha_2 \boldsymbol{u}_2) && (7) \\ &= \boldsymbol{u}_1 + \alpha_2 \lambda_2 \boldsymbol{u}_2 && (8) \end{aligned}$$

Plugging in $\lambda_2 = c$, we see that

$$\boldsymbol{x}^{(k)} = \boldsymbol{u}_1 + \alpha_2 c \boldsymbol{u}_2 \tag{9}$$

This allows us to solve for $\boldsymbol{u}_1$ in closed form:

$$\boldsymbol{u}_1 = \frac{\boldsymbol{x}^{(k)} - c\boldsymbol{x}^{(k-1)}}{1-c} \tag{10}$$

Results and Discussion Figure 2 shows the convergence of Simple Extrapolation and the standard Power Method, where there was one application of Simple Extrapolation at iteration 3 of the Power Method. Simple Extrapolation is not effective, as the assumption that c is the only eigenvalue of modulus c is inaccurate. In fact, by doubling the error in the eigenspace corresponding to eigenvalue $-c$, this extrapolation technique *slows down* the convergence of the Power Method.

5.2 A^2 Extrapolation

Formulation The next extrapolation rule assumes that the iterate $\boldsymbol{x}^{(k-2)}$ can be expressed as a linear combination of the eigenvectors u_1, u_2, and u_3, where u_2 has eigenvalue c and u_3 has eigenvalue $-c$.

$$\boldsymbol{x}^{(k-2)} = \boldsymbol{u}_1 + \alpha_2 \boldsymbol{u}_2 + \alpha_3 \boldsymbol{u}_3 \tag{11}$$

Now consider the current iterate $\boldsymbol{x}^{(k)}$; because the Power Method generates iterates by successive multiplication by A, we can write $\boldsymbol{x}^{(k)}$ as

$$\begin{aligned} \boldsymbol{x}^{(k)} &= A^2\boldsymbol{x}^{(k-2)} && (12) \\ &= A^2(\boldsymbol{u}_1 + \alpha_2 \boldsymbol{u}_2 + \alpha_3 \boldsymbol{u}_3) && (13) \\ &= \boldsymbol{u}_1 + \alpha_2 \lambda_2^2 \boldsymbol{u}_2 + \alpha_2 \lambda_3^2 \boldsymbol{u}_3 && (14) \end{aligned}$$

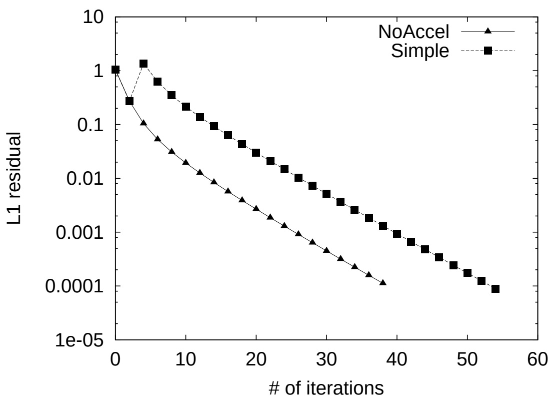

Fig. 2. Comparison of convergence rates for Power Method and Simple Extrapolation on LargeWeb for $c = 0.85$.

Plugging in $\lambda_2 = c$ and $\lambda_3 = -c$, we see that

$$\boldsymbol{x}^{(k)} = \boldsymbol{u}_1 + c^2(\alpha_2 \boldsymbol{u}_2 + \alpha_3 \boldsymbol{u}_3) \tag{15}$$

This allows us to solve for $\boldsymbol{u}_1$ in closed form:

$$\boldsymbol{u}_1 = \frac{\boldsymbol{x}^{(k)} - c^2 \boldsymbol{x}^{(k-2)}}{1 - c^2} \tag{16}$$

A^2 Extrapolation eliminates error along the eigenspaces corresponding to eigenvalues of c and $-c$.

Results and Discussion Figure 3 shows the convergence of A^2 extrapolated PageRank and the standard Power Method where A^2 Extrapolation was applied once at iteration 4. Empirically, A^2 extrapolation speeds up the convergence of the Power Method by 18%. The initial effect of the application increases the residual, but by correctly subtracting off much of the largest non-principal eigenvectors, the convergence upon further iterations of the Power Method is sped up.

5.3 A^d Extrapolation

Formulation The previous extrapolation rule made use of the fact that $(-c)^2 = c^2$. We can generalize that derivation to the case where the eigenvalues of modulus c are given by cd_i, where $\{d_i\}$ are the dth roots of unity. From Theorem 2.1 of [3] and Theorem 1 given in the Appendix, it follows that these eigenvalues arise from leaf nodes in the strongly connected component (SCC) graph of the Web that are cycles of length d. Because we know empirically that the Web has such leaf nodes in the SCC graph, it is likely that eliminating error along the dimensions of eigenvectors corresponding to these eigenvalues will speed up PageRank.

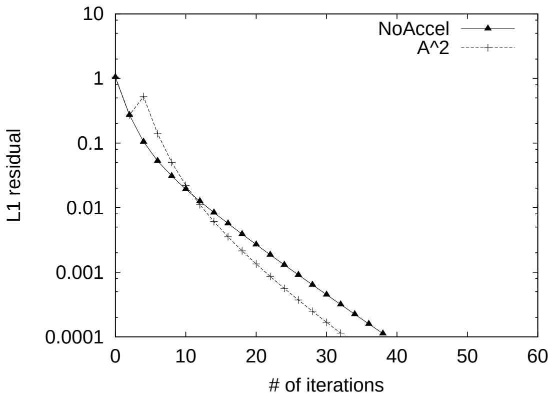

Fig. 3. Comparison of convergence rates for Power Method and A^2 Extrapolation on LargeWeb for $c = 0.85$.

We make the assumption that $\boldsymbol{x}^{(k-d)}$ can be expressed as a linear combination of the eigenvectors $\{u_1 \ldots u_{d+1}\}$, where the eigenvalues of $\{u_2 \ldots u_{d+1}\}$ are the dth roots of unity, scaled by c.

$$\boldsymbol{x}^{(k-d)} = \boldsymbol{u}_1 + \sum_{i=2}^{d+1} \alpha_i \boldsymbol{u}_i \tag{17}$$

Then consider the current iterate $\boldsymbol{x}^{(k)}$; because the Power Method generates iterates by successive multiplication by A, we can write $\boldsymbol{x}^{(k)}$ as

$$\begin{aligned}\boldsymbol{x}^{(k)} &= A^d \boldsymbol{x}^{(k-d)} && (18)\\ &= A^d(\boldsymbol{u}_1 + \sum_{i=2}^{d+1} \alpha_i \boldsymbol{u}_i) && (19)\\ &= \boldsymbol{u}_1 + \sum_{i=2}^{d+1} \alpha_i \lambda_i^d \boldsymbol{u}_i && (20)\\ & && (21)\end{aligned}$$

But since λ_i is cd_i, where d_i is a dth root of unity,

$$\begin{aligned}\boldsymbol{x}^{(k)} &= \boldsymbol{u}_1 + c^d \sum_{i=2}^{d+1} \alpha_i \boldsymbol{u}_i && (22)\\ & && (23)\end{aligned}$$

This allows us to solve for $\boldsymbol{u}_1$ in closed form:

$$\boldsymbol{u}_1 = \frac{\boldsymbol{x}^{(k)} - c^d \boldsymbol{x}^{(k-d)}}{1 - c^d} \tag{24}$$

```
function $\boldsymbol{x}^* = \mathit{PowerExtrapolation}(\boldsymbol{x}^{(k-d)}, \boldsymbol{x}^{(k)})$ {
$\boldsymbol{x}^* = (\boldsymbol{x}^{(k)} - c^d\boldsymbol{x}^{(k-d)})(1 - c^d)^{-1}$;
}
```

Algorithm 3: Power Extrapolation

```
function $\boldsymbol{x}^{(n)} = \mathit{ExtrapolatedPowerMethod}(d)$ {
$\boldsymbol{x}^{(0)} = \boldsymbol{v}$;
$k = 1$;
repeat
    $\boldsymbol{x}^{(k)} = A\boldsymbol{x}^{(k-1)}$;
    $\delta = ||x^{(k)} - x^{(k-1)}||_1$;
    if $k == d + 2$,
        $\boldsymbol{x}^{(k)} = PowerExtrapolation(\boldsymbol{x}^{(k-d)}, \boldsymbol{x}^{(k)})$;
    $k = k + 1$;
until $\delta < \epsilon$;
}
```

Algorithm 4: Power Method with Power Extrapolation

For instance, for $d = 4$, the assumption made is that the nonprincipal eigenvalues of modulus c are given by c, $-c$, ci, and $-ci$ (i.e., the 4th roots unity). A graph in which the leaf nodes in the SCC graph contain only cycles of length l, where l is any divisor of $d = 4$ has exactly this property.

Algorithms 3 and 4 show how to use A^d Extrapolation in conjunction with the Power Method. Note that Power Extrapolation with $d = 1$ is just Simple Extrapolation.

Operation Count The overhead in performing the extrapolation shown in Algorithm 3 comes from computing the linear combination $(\boldsymbol{x}^{(k)} - c^d\boldsymbol{x}^{(k-d)})(1 - c^d)^{-1}$, an $O(n)$ computation.

In our experimental setup, the overhead of a single application of Power Extrapolation is 1% the cost of a standard power iteration. Furthermore, Power Extrapolation needs to be applied only once to achieve the full benefit.

Results and Discussion In our experiments, A^d Extrapolation performs the best for $d = 6$. Figure 4 plots the convergence of A^d Extrapolation for $d \in \{1, 2, 4, 6, 8\}$, as well as of the standard Power Method, for $c = 0.85$ and $c = 0.90$.

The wallclock speedups, compared with the standard Power Method, for these 5 values of d for $c = 0.85$ are given in Table 1.

For comparison, Figure 5 compares the convergence of the Quadratic Extrapolated PageRank with A^6 Extrapolated PageRank. Note that the speedup in convergence is similar; however, A^6 Extrapolation is much simpler to implement, and has negligible overhead, so that its wallclock-speedup is higher. In particular, each application of Quadratic Extrapolation requires 32% of the cost of an iteration, and must be applied several times to achieve maximum benefit.

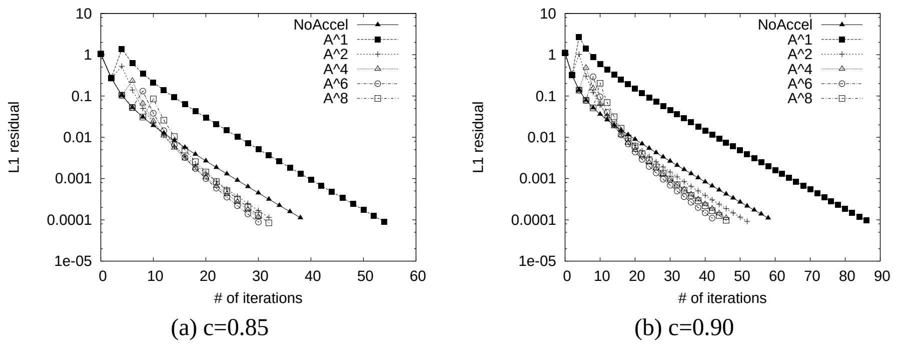

Fig. 4. Convergence rates for A^d Extrapolation, for $d \in \{1, 2, 4, 6, 8\}$, compared with standard Power Method.

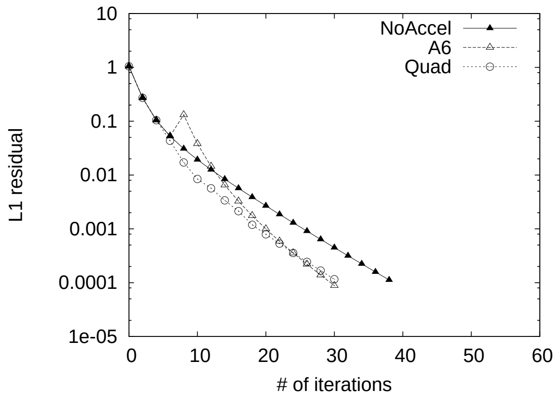

Fig. 5. Comparison of convergence rates for Power Method, A^6 Extrapolation, and Quadratic Extrapolation on LARGEWEB for $c = 0.85$.

6 Acknowledgements

This paper is based on work supported in part by the National Science Foundation under Grants No. IIS-0085896 and CCR-9971010, and in part by the Research Collaboration between NTT Communication Science Laboratories, Nippon Telegraph and Telephone Corporation and CSLI, Stanford University (research project on Concept Bases for Lexical Acquisition and Intelligently Reasoning with Meaning).

Table 1. Wallclock speedups for A^dExtrapolation, for $d \in 1, 2, 4, 6, 8$, and Quadratic Extrapolation

Type	speedup
$d = 1$	-28%
$d = 2$	18%
$d = 4$	25.8%
$d = 6$	30%
$d = 8$	21.8%
Quadratic	20.8%

References

1. G. Grimmett and D. Stirzaker. *Probability and Random Processes*. Oxford University Press, 1989.
2. T. H. Haveliwala. Topic-sensitive PageRank. In *Proceedings of the Eleventh International World Wide Web Conference*, 2002.
3. T. H. Haveliwala and S. D. Kamvar. The second eigenvalue of the Google matrix. *Stanford University Technical Report*, 2003.
4. J. Hirai, S. Raghavan, H. Garcia-Molina, and A. Paepcke. WebBase: A repository of web pages. In *Proceedings of the Ninth International World Wide Web Conference*, 2000.
5. D. L. Isaacson and R. W. Madsen. *Markov Chains: Theory and Applications*, chapter IV, pages 126–127. John Wiley and Sons, Inc., New York, 1976.
6. G. Jeh and J. Widom. Scaling personalized web search. In *Proceedings of the Twelfth International World Wide Web Conference*, 2003.
7. S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub. Extrapolation methods for accelerating PageRank computations. In *Proceedings of the Twelfth International World Wide Web Conference*, 2003.
8. J. Kleinberg, S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. The web as a graph: Measurements, models, and methods. In *Proceedings of the International Conference on Combinatorics and Computing*, 1999.
9. L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. *Stanford Digital Libraries Working Paper*, 1998.
10. M. Richardson and P. Domingos. The intelligent surfer: Probabilistic combination of link and content information in PageRank. In *Advances in Neural Information Processing Systems*, volume 14. MIT Press, Cambridge, MA, 2002.

Appendix

This appendix repeats Theorem IV.2.5 from [5].

Theorem 1. *(Theorem IV.2.5 from [5]) If P is the transition matrix of an irreducible periodic Markov chain with period d, then the dth roots of unity are eigenvalues of P. Further, each of these eigenvalues is of multiplicity one and there are no other eigenvalues of modulus 1.*