PageRank Modeling using Markov Chains

Tomas Etlin

February 3, 2025

Abstract

Ranking and information retrieval play a critical role in structuring vast networks, from web search engines to financial markets. One fundamental challenge is determining the relative importance of nodes in a network, where connections follow complex, probabilistic patterns. In particular, link structures influence node significance, making them an essential component in applications such as search algorithms, recommendation systems, and network analysis.

In this project, I develop a simplified PageRank algorithm using Markov Chains to model the probability distribution of a web surfer navigating a network of linked web pages. I construct an adjacency graph from sampled web data, build transition probability matrices, and handle key challenges such as dangling nodes and isolated clusters. Two approaches—an eigenvector-based stationary distribution and an iterative power method—are implemented and compared in terms of computational complexity and convergence behavior. By analyzing PageRank scores across different damping factors and network sizes, I assess the sensitivity of rankings to structural variations. My results highlight the efficiency of iterative methods for large-scale networks and demonstrate the impact of link topology on ranking stability.

1. Introduction: Problem Overview

In this project, I aim to develop a simplified PageRank algorithm through Markov Chains to assess the relative significance of web pages within a sample network segment. This model simulates a web surfer's navigation through links on various pages, thereby ranking each page according to its likelihood of being visited. By starting from a specific webpage, I collect data to construct an adjacency graph of connected pages, define transition probabilities between these pages, and address situations where navigation may become "stuck."

Data Collection and Adjacency Graph Construction

To start, I selected https://www.gradescope.com/ as the initial page. Using the `surfer.m` script, I captured networks of varying sizes with $N = 100, 200, 300, 400$, and 500 linked pages. This resulted in:

- **List of URLs (U)**: Pages visited in each crawl.
- **Adjacency Matrix (G)**: Binary matrix indicating link presence, where $G_{ij} = 1$ if page j links to page i and $G_{ij} = 0$ otherwise.

2. Transition Probability Matrix Construction

Constructing a transition probability matrix from the adjacency matrix G requires addressing key challenges in web navigation:

Handling Dangling Nodes

Pages without outgoing links (dangling nodes) can trap the surfer on a single page, disrupting continuous movement through the network. To address this, I adjusted the matrix by setting each column of the transition matrix G corresponding to a dangling node to 1. This adjustment allows us to normalize G without altering the network's structure, as follows:

```
% Handle dangling nodes by setting entire column in G to 1
dangling_nodes = find(sum(G, 1) == 0);
G(:, dangling_nodes) = 1;

% Normalize columns of G to construct P_original
column_sums = sum(G, 1);
P_original = G ./ column_sums;

% Loop over different p values
for p_idx = 1:num_p
```

In this approach, `P_original` is the normalized transition matrix. For each page, the transition probability is calculated by dividing each entry by the sum of its column. By setting dangling node columns to 1, we ensure that each page receives a uniform probability distribution in the presence of dangling nodes, maintaining a balanced surfer flow across the network.

Ensuring Irreducibility with the Random Surfer Model

The web graph might contain isolated clusters, limiting reachability across pages. To address this, I applied the "random surfer" model, assigning a probability p for following links on the current page and $1 - p$ for randomly jumping to any page. The combined transition probability matrix is:

$$P = p \times P_{\text{original}} + (1 - p) \times \frac{1}{N} \times \mathbf{E}$$

where:

- p is the damping factor.
- $\mathbf{E}$ is an $N \times N$ matrix of ones.
- $\frac{1}{N}$ represents equal probability of jumping to any page.

Code implementation:

```
P = p * P_original + (1 - p) * (1 / N_actual) * ones(N_actual, N_actual);
```

3. Calculation of the Stationary Probability Vector

With the transition matrix P, I calculate the stationary probability vector, indicating the probability distribution of the surfer's location over time.

Method 1: Eigenvector Approach

The eigenvector method calculates the stationary distribution by finding the dominant eigenvector of P using MATLAB's `eig` function. This eigenvector represents the PageRank by normalizing it to sum to 1.

Code implementation:

```
%% Method 1: Eigenvector Methode
tic; % Start timing
[V, D] = eig(P);
% Find the eigenvector corresponding to the eigenvalue closest to 1
[~, idx_eig] = min(abs(diag(D) - 1));
principal_eigenvector = abs(real(V(:, idx_eig)));
% Normalize the eigenvector
pagerank_eig = principal_eigenvector / sum(principal_eigenvector);
```

```
timings_eig(n_idx, p_idx) = toc; % Record timing

% Extract top 10 pages
[prob_eig, indices_eig] = sort(pagerank_eig, ’descend’);
top_pages_eig{n_idx, p_idx}
= [U(indices_eig(1:min(10, length(U)))), num2cell(prob_eig(1:min(10, length(U))))];
```

Eigenvector Method Complexity Analysis

The computational complexity of the eigenvector method is outlined by examining each section of the code:

1. **Eigenvector Calculation** (`[V, D] = eig(P)`):
   - Here, the function `eig(P)` computes all eigenvalues and eigenvectors of the matrix P.
   - Since P is a dense $N \times N$ matrix, computing its eigenvalues and eigenvectors has a time complexity of $O(N^3)$.
   - To understand why eigenvalue computation for a dense $N \times N$ matrix has a complexity of $O(N^3)$, we examine the matrix multiplication process.
     - **Matrix Multiplication Method**: When multiplying two $N \times N$ matrices, each entry in the resulting matrix is calculated by taking the dot product of a row from the first matrix and a column from the second, requiring N^2 such entries to be computed (since there are N rows and N columns).
     - **Dot Product computation**: Each dot product requires N multiplications and $N - 1$ additions, totaling approximately $O(N)$ operations for each entry.

   Since we need $O(N)$ operations per entry and there are N^2 entries in the result, the overall complexity for multiplying two $N \times N$ matrices becomes:

   Complexity: $O(N^2) \times O(N) = O(N^3)$

2. **Finding the Principal Eigenvector**:
   - The line `[ , idx_eig] = min(abs(diag(D) - 1))` identifies the index of the eigenvalue closest to 1, which corresponds to the principal eigenvector.
   - Extracting this index requires examining each entry of the diagonal matrix D, which has a complexity of $O(N)$.
   - Once the index is found, `V(:, idx_eig)` extracts the corresponding eigenvector.
   - **Complexity**: $O(N)$.

3. **Normalization** (`pagerank_eig = principal_eigenvector / sum(principal_eigenvector)`):
   - This step normalizes the principal eigenvector so that the sum of its entries equals 1.
   - Calculating the sum and normalizing each entry takes $O(N)$.
   - **Complexity**: $O(N)$.

Total Complexity of the Eigenvector Method

The dominant operation in the eigenvector method is calculating all eigenvalues and eigenvectors, which has a complexity of $O(N^3)$. Thus, the overall complexity for this method is:

$$O(N^3)$$

Method 2: Power Method

Starting with an initial probability vector focused on the starting URL, I iteratively update probabilities until they converge, using:

$$\text{rank}_{\text{new}} = P \cdot \text{rank}$$

$$\text{rank}_{\text{new}} = \frac{\text{rank}_{\text{new}}}{\sum \text{rank}_{\text{new}}}$$

$$\text{if} \quad \|\text{rank}_{\text{new}} - \text{rank}\|_1 < \text{tolerance}, \quad \text{stop.}$$

$$\text{else, set rank} = \text{rank}_{\text{new}} \quad \text{and continue.}$$

I continue iterations until the probability difference falls below 1×10^{-6} = tolerance.

Code implementation:

```
%% Method 2: Power Method
tic; % Start timing
% Initial probability distribution: uniform over all nodes
rank = ones(N_actual, 1) / N_actual;

% Power method parameters
tolerance = 1e-6;
max_iterations = 1000;
for iter = 1:max_iterations
    rank_new = P * rank;
    % Normalize the rank vector
    rank_new = rank_new / sum(rank_new);
    % Check convergence
    if norm(rank_new - rank, 1) < tolerance
        break;
    end
    rank = rank_new;
end
pagerank_power = rank;
timings_power(n_idx, p_idx) = toc; % Record timing
iterations_power(n_idx, p_idx) = iter; % Record number of iterations

% Extract top 10 pages
[prob_power, indices_power] = sort(pagerank_power, 'descend');
top_pages_power{n_idx, p_idx}
= [U(indices_power(1:min(10, length(U)))), num2cell(prob_power(1:min(10, length(U))))];
```

Power Method Complexity Analysis

The computational complexity of the power method can be understood by examining each component of the code:

1. **Initialization**:
   - The initial probability distribution `rank` is set as a uniform distribution over all nodes.
   - **Complexity**: Initializing `rank` takes $O(N)$, where N is the number of nodes.
2. **Main Loop** (`for iter = 1:max_iterations`):

- The loop runs a maximum of `max_iterations` iterations (e.g., 1000). However, the loop will break early if convergence is achieved based on the defined `tolerance`.

3. **Matrix-Vector Multiplication** (`rank_new = P * rank`):

   - In each iteration, `rank_new` is updated by multiplying the transition matrix P by the current rank vector.
   - Since P includes a stochastic adjustment ensuring irreducibility:

     $$P = pP_{\text{original}} + (1-p)\frac{1}{N}\mathbf{1}\mathbf{1}^T$$

     it is fully dense rather than sparse.
   - **Complexity per iteration**: $O(N^2)$ due to matrix-vector multiplication on a dense matrix.

4. **Normalization** (`rank_new = rank_new / sum(rank_new)`):

   - This step ensures the rank vector sums to 1, involving summing `rank_new` and dividing each element by the sum.
   - **Complexity**: $O(N)$.

5. **Convergence Check** (`if norm(rank_new - rank, 1) < tolerance`):

   - This check calculates the 1-norm difference between `rank_new` and `rank`, comparing it to `tolerance` to determine convergence.
   - **Complexity**: $O(N)$.

6. **Updating** `rank`:

   - If convergence hasn't been reached, `rank` is updated to `rank_new` for the next iteration.
   - **Complexity**: $O(N)$.

Total Complexity of the Power Method

The dominant step is the matrix-vector multiplication, which is $O(N^2)$ per iteration.

Let k represent the number of iterations until convergence. Thus, the overall complexity of the power method is:

$$O(N^2 \times k)$$

4. Results for Different Network Sizes (N) at $p = 0.05$

The top 10 pages and their probabilities for each N are presented below.

Eigenvector Method Results

N	Rank	Page (Probability)
N = 100		
	1	`https://schema.org` (0.016126)
	2	`https://twitter.com/gradescope` (0.010689)
	3	`https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion` (0.010662)
	4	`https://www.facebook.com/TurnitinSoftware` (0.010629)
	5	`http://schema.org/Article` (0.010476)
	6	`https://www.youtube.com/creators` (0.010420)
	7	`https://www.turnitin.com` (0.010372)

N	Rank	Page (Probability)
	8	https://www.linkedin.com/company/turnitin (0.010357)
	9	https://tv.youtube.com/learn/nflsundayticket (0.010343)
	10	https://github.com/gradescope/autograder_samples/edit/master/docs/index.md (0.010258)
N = 200		
	1	https://schema.org (0.008241)
	2	https://www.facebook.com/TurnitinSoftware (0.005457)
	3	https://twitter.com/gradescope (0.005414)
	4	https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion (0.005382)
	5	https://www.turnitin.com (0.005325)
	6	https://www.linkedin.com/company/turnitin (0.005318)
	7	https://www.youtube.com/creators (0.005254)
	8	https://www.mkdocs.org (0.005253)
	9	https://github.com/readthedocs/sphinx_rtd_theme (0.005253)
	10	https://tv.youtube.com/learn/nflsundayticket (0.005215)
N = 300		
	1	https://schema.org (0.004558)
	2	https://turnitin.statuspage.io (0.004209)
	3	http://schema.org (0.004063)
	4	https://supportcenter.turnitin.com/s (0.003869)
	5	https://www.turnitin.com (0.003867)
	6	https://validator.schema.org (0.003859)
	7	https://www.facebook.com/TurnitinSoftware (0.003644)
	8	https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion (0.003583)
	9	https://www.linkedin.com/company/turnitin (0.003551)
	10	https://twitter.com/gradescope (0.003524)
N = 400		
	1	https://schema.org (0.003403)
	2	http://schema.org (0.003224)
	3	https://turnitin.statuspage.io (0.003162)
	4	https://validator.schema.org (0.002916)
	5	https://supportcenter.turnitin.com/s (0.002890)
	6	https://www.turnitin.com (0.002867)
	7	https://www.linkedin.com/company/turnitin (0.002768)
	8	https://www.facebook.com/TurnitinSoftware (0.002729)
	9	https://twitter.com/turnitin (0.002714)
	10	https://www.pinterest.com (0.002712)
N = 500		
	1	https://schema.org (0.003009)
	2	http://schema.org (0.002876)
	3	https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion (0.002843)
	4	https://turnitin.statuspage.io (0.002466)
	5	https://www.turnitin.com (0.002248)
	6	https://supportcenter.turnitin.com/s (0.002246)
	7	https://www.linkedin.com/company/turnitin (0.002214)
	8	https://fonts.gstatic.com (0.002202)
	9	https://www.facebook.com/TurnitinSoftware (0.002183)
	10	https://www.youtube.com/creators (0.002179)

Power Method Results

N	Rank	Page (Probability)
N = 100		
	1	`https://schema.org` (0.016126)
	2	`https://twitter.com/gradescope` (0.010689)
	3	`https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion` (0.010662)
	4	`https://www.facebook.com/TurnitinSoftware` (0.010629)
	5	`http://schema.org/Article` (0.010476)
	6	`https://www.youtube.com/creators` (0.010420)
	7	`https://www.turnitin.com` (0.010372)
	8	`https://www.linkedin.com/company/turnitin` (0.010357)
	9	`https://tv.youtube.com/learn/nflsundayticket` (0.010343)
	10	`https://github.com/gradescope/autograder_samples/edit/master/docs/index.md` (0.010258)
N = 200		
	1	`https://schema.org` (0.008241)
	2	`https://www.facebook.com/TurnitinSoftware` (0.005457)
	3	`https://twitter.com/gradescope` (0.005414)
	4	`https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion` (0.005382)
	5	`https://www.turnitin.com` (0.005325)
	6	`https://www.linkedin.com/company/turnitin` (0.005318)
	7	`https://www.youtube.com/creators` (0.005254)
	8	`https://www.mkdocs.org` (0.005253)
	9	`https://github.com/readthedocs/sphinx_rtd_theme` (0.005253)
	10	`https://tv.youtube.com/learn/nflsundayticket` (0.005215)
N = 300		
	1	`https://schema.org` (0.004558)
	2	`https://turnitin.statuspage.io` (0.004209)
	3	`http://schema.org` (0.004063)
	4	`https://supportcenter.turnitin.com/s` (0.003869)
	5	`https://www.turnitin.com` (0.003867)
	6	`https://validator.schema.org` (0.003859)
	7	`https://www.facebook.com/TurnitinSoftware` (0.003644)
	8	`https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion` (0.003583)
	9	`https://www.linkedin.com/company/turnitin` (0.003551)
	10	`https://twitter.com/gradescope` (0.003524)
N = 400		
	1	`https://schema.org` (0.003403)
	2	`http://schema.org` (0.003224)
	3	`https://turnitin.statuspage.io` (0.003162)
	4	`https://validator.schema.org` (0.002916)
	5	`https://supportcenter.turnitin.com/s` (0.002890)
	6	`https://www.turnitin.com` (0.002867)
	7	`https://www.linkedin.com/company/turnitin` (0.002768)
	8	`https://www.facebook.com/TurnitinSoftware` (0.002729)
	9	`https://twitter.com/turnitin` (0.002714)
	10	`https://www.pinterest.com` (0.002712)
N = 500		
	1	`https://schema.org` (0.003009)
	2	`http://schema.org` (0.002876)
	3	`https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion` (0.002843)
	4	`https://turnitin.statuspage.io` (0.002466)
	5	`https://www.turnitin.com` (0.002248)
	6	`https://supportcenter.turnitin.com/s` (0.002246)
	7	`https://www.linkedin.com/company/turnitin` (0.002214)

N	Rank	Page (Probability)
	8	https://fonts.gstatic.com (0.002202)
	9	https://www.facebook.com/TurnitinSoftware (0.002183)
	10	https://www.youtube.com/creators (0.002179)

Performance Metrics

N	Eigenvector Time (s)	Power Method Time (s)	Power Iterations
100	0.0298	0.0015	5
200	0.0093	0.0001	5
300	0.0185	0.0001	5
400	0.0372	0.0001	4
500	0.0616	0.0002	4

Analysis of Results with Low Damping Factor $p = 0.05$

With a low damping factor $p = 0.05$, the PageRank algorithm heavily emphasizes random jumps rather than following links within the network. This means that only 5% of the probability mass is applied to link-following behavior, while 95% is dedicated to jumping randomly to any page. As a result, the influence of the link structure is significantly reduced, and PageRank scores become more uniform across the network, making the random surfer model behave closer to a uniform distribution.

For example, while schema.org generally ranks highly due to its extensive inbound links, its score is dampened here because the high random jump probability limits the impact of those links. Social media pages, like Twitter and Facebook, which also benefit from extensive linking within and beyond the network, maintain high ranks but with reduced scores compared to settings with a higher p value. In this configuration, even highly connected pages don't achieve the pronounced rankings they might in a typical PageRank setting with higher p values.

Computational Efficiency: Eigenvector vs. Power Method with Low p

The low damping factor $p = 0.05$ also influences computational efficiency differently for the eigenvector and power methods. In the eigenvector method, the low p value reduces the dominance of link-following in the transition matrix, making the matrix closer to a uniform matrix. This reduces the structure that can be leveraged in eigenvalue decomposition, increasing computational complexity. As a result, the eigenvector method, which directly computes the dominant eigenvector via eigenvalue decomposition with a time complexity of $O(N^3)$, becomes more computationally intensive as it handles a matrix that lacks strong hierarchical structure.

In contrast, the power method remains efficient in this low p setting. Since the power method iteratively applies the transition matrix to approximate the PageRank vector, it benefits from the matrix's sparsity, achieving $O(N)$ complexity per iteration. The high random jump probability leads to quicker convergence, as fewer iterations are needed for stabilization when the influence of specific links is limited. This efficiency is especially pronounced in sparse matrices, making the power method well-suited for large-scale applications.

Effects of Increasing Network Size from $N = 100$ to $N = 500$

As network size N increases from 100 to 500, individual PageRank scores for top-ranked pages decrease. This is due to the fixed probability mass of 1, which is distributed across more pages as N grows, naturally reducing each page's individual share of the score. Despite this, schema.org and social media pages consistently rank highly across all network sizes, though with lower absolute scores as the network expands.

In terms of computational time, the difference between the eigenvector and power methods becomes more pronounced as N increases. The power method's reliance on sparse structure and iterative approximation maintains efficient performance even as N grows, as its $O(N)$ complexity per iteration scales well with network size. In contrast, the eigenvector method's $O(N^3)$ complexity becomes increasingly challenging with larger N, especially with the low p value reducing the matrix's structure.

The low damping factor also affects convergence as N increases. With $p = 0.05$, the random jumps help the power method achieve fast convergence, as the algorithm does not need to capture a strong link-following pattern, which would require more iterations to stabilize. In larger networks, this effect remains consistent, with the power method converging in 4-5 iterations across all network sizes, while the eigenvector method becomes more resource-intensive with size.

In summary, for low p values, the power method outperforms the eigenvector method due to its ability to handle sparse structures and achieve quick convergence. Schema.org and social media pages maintain top rankings across all network sizes, despite slightly reduced scores, as their central roles and extensive linkage remain influential even when random jumps dominate. This analysis highlights the power method's advantages in efficiency and scalability, making it the preferred approach for large, sparse networks under the random surfer model.

5. Results for Different Network Sizes (N) at $p = 0.10$

The top 10 pages and their probabilities for each N are presented below.

Eigenvector Method Results

N	Rank	Page (Probability)
N = 100		
	1	`https://schema.org` (0.022945)
	2	`https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion` (0.011394)
	3	`https://twitter.com/gradescope` (0.011333)
	4	`https://www.facebook.com/TurnitinSoftware` (0.011316)
	5	`http://schema.org/Article` (0.010902)
	6	`https://www.youtube.com/creators` (0.010844)
	7	`https://www.turnitin.com` (0.010732)
	8	`https://www.linkedin.com/company/turnitin` (0.010721)
	9	`https://tv.youtube.com/learn/nflsundayticket` (0.010702)
	10	`https://github.com/gradescope/autograder_samples/edit/master/docs/index.md` (0.010534)
N = 200		
	1	`https://schema.org` (0.011937)
	2	`https://www.facebook.com/TurnitinSoftware` (0.005969)
	3	`https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion` (0.005819)
	4	`https://twitter.com/gradescope` (0.005802)
	5	`https://www.turnitin.com` (0.005658)
	6	`https://www.linkedin.com/company/turnitin` (0.005654)
	7	`https://www.mkdocs.org` (0.005525)
	8	`https://github.com/readthedocs/sphinx_rtd_theme` (0.005525)
	9	`https://www.youtube.com/creators` (0.005517)
	10	`https://tv.youtube.com/learn/nflsundayticket` (0.005445)
N = 300		
	1	`https://schema.org` (0.005931)
	2	`https://turnitin.statuspage.io` (0.005035)
	3	`http://schema.org` (0.004811)
	4	`https://validator.schema.org` (0.004520)
	5	`https://supportcenter.turnitin.com/s` (0.004463)
	6	`https://www.turnitin.com` (0.004389)
	7	`https://www.facebook.com/TurnitinSoftware` (0.003991)
	8	`https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion` (0.003861)
	9	`https://www.linkedin.com/company/turnitin` (0.003781)
	10	`http://purl.org/dc/elements/1.1` (0.003708)

N	Rank	Page (Probability)
N = 400		
	1	https://schema.org (0.004430)
	2	http://schema.org (0.003959)
	3	https://turnitin.statuspage.io (0.003790)
	4	https://validator.schema.org (0.003444)
	5	https://supportcenter.turnitin.com/s (0.003326)
	6	https://www.turnitin.com (0.003227)
	7	https://www.linkedin.com/company/turnitin (0.003065)
	8	https://www.facebook.com/TurnitinSoftware (0.002987)
	9	https://twitter.com/turnitin (0.002954)
	10	http://community.forumbee.com/t/18hzyj/enabling-browser-cookies (0.002937)
N = 500		
	1	https://schema.org (0.004076)
	2	http://schema.org (0.003759)
	3	https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion (0.003688)
	4	https://turnitin.statuspage.io (0.002908)
	5	https://supportcenter.turnitin.com/s (0.002525)
	6	https://www.turnitin.com (0.002489)
	7	https://www.linkedin.com/company/turnitin (0.002451)
	8	https://github.com/schemaorg/schemaorg (0.002410)
	9	https://validator.schema.org (0.002404)
	10	https://fonts.gstatic.com (0.002396)

Power Method Results

N	Rank	Page (Probability)
N = 100		
	1	https://schema.org (0.022945)
	2	https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion (0.011394)
	3	https://twitter.com/gradescope (0.011333)
	4	https://www.facebook.com/TurnitinSoftware (0.011316)
	5	http://schema.org/Article (0.010902)
	6	https://www.youtube.com/creators (0.010844)
	7	https://www.turnitin.com (0.010732)
	8	https://www.linkedin.com/company/turnitin (0.010721)
	9	https://tv.youtube.com/learn/nflsundayticket (0.010702)
	10	https://github.com/gradescope/autograder_samples/edit/master/docs/index.md (0.010534)
N = 200		
	1	https://schema.org (0.011937)
	2	https://www.facebook.com/TurnitinSoftware (0.005969)
	3	https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion (0.005819)
	4	https://twitter.com/gradescope (0.005802)
	5	https://www.turnitin.com (0.005658)
	6	https://www.linkedin.com/company/turnitin (0.005654)
	7	https://www.mkdocs.org (0.005525)
	8	https://github.com/readthedocs/sphinx_rtd_theme (0.005525)
	9	https://www.youtube.com/creators (0.005517)
	10	https://tv.youtube.com/learn/nflsundayticket (0.005445)
N = 300		
	1	https://schema.org (0.005931)
	2	https://turnitin.statuspage.io (0.005035)
	3	http://schema.org (0.004811)

N	Rank	Page (Probability)
	4	`https://validator.schema.org` (0.004520)
	5	`https://supportcenter.turnitin.com/s` (0.004463)
	6	`https://www.turnitin.com` (0.004389)
	7	`https://www.facebook.com/TurnitinSoftware` (0.003991)
	8	`https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion` (0.003861)
	9	`https://www.linkedin.com/company/turnitin` (0.003781)
	10	`http://purl.org/dc/elements/1.1` (0.003708)
N = 400		
	1	`https://schema.org` (0.004430)
	2	`http://schema.org` (0.003959)
	3	`https://turnitin.statuspage.io` (0.003790)
	4	`https://validator.schema.org` (0.003444)
	5	`https://supportcenter.turnitin.com/s` (0.003326)
	6	`https://www.turnitin.com` (0.003227)
	7	`https://www.linkedin.com/company/turnitin` (0.003065)
	8	`https://www.facebook.com/TurnitinSoftware` (0.002987)
	9	`https://twitter.com/turnitin` (0.002954)
	10	`http://community.forumbee.com/t/18hzyj/enabling-browser-cookies` (0.002937)
N = 500		
	1	`https://schema.org` (0.004076)
	2	`http://schema.org` (0.003759)
	3	`https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion` (0.003688)
	4	`https://turnitin.statuspage.io` (0.002908)
	5	`https://supportcenter.turnitin.com/s` (0.002525)
	6	`https://www.turnitin.com` (0.002489)
	7	`https://www.linkedin.com/company/turnitin` (0.002451)
	8	`https://github.com/schemaorg/schemaorg` (0.002410)
	9	`https://validator.schema.org` (0.002404)
	10	`https://fonts.gstatic.com` (0.002396)

Performance Metrics

N	Eigenvector Time (s)	Power Method Time (s)	Power Iterations
100	0.0032	0.0002	6
200	0.0096	0.0001	6
300	0.0179	0.0001	6
400	0.0397	0.0002	6
500	0.0593	0.0002	6

Analysis of Results with Damping Factor $p = 0.10$

At $p = 0.10$, both the eigenvector and power methods continue to yield consistent PageRank values across varying network sizes N, confirming convergence to the same stationary distribution. However, the slight increase in p from 0.05 to 0.10 reduces the frequency of random jumps and increases the influence of the link-following structure. This shift slightly amplifies the PageRank values of well-connected pages, reflecting a closer alignment with network connectivity.

Impact on Top-Ranked Pages

As observed, schema.org remains the highest-ranked page, while social media and technical resources—such as Twitter, Facebook, and schema validation pages—continue to hold top positions. With $p = 0.10$, these pages experience a small increase in their PageRank scores compared to $p = 0.05$. The higher damping factor accentuates the importance of the network's link structure, benefiting pages with numerous inbound links. Consequently, schema.org's role as a hub for structured data, and the widespread linking to social media pages, results in higher relative scores at this p level due to the increased likelihood of link-following.

Comparing Computational Efficiency Between Methods

For $p = 0.10$, the eigenvector method's computation time remains influenced by its $O(N^3)$ complexity, as it relies on eigenvalue decomposition. This computational load is manageable at small N but grows quickly with larger networks. The power method, with its $O(N)$ complexity per iteration, continues to be more efficient, especially leveraging the sparse transition matrix. At $p = 0.10$, the power method converges slightly slower, typically within 6 iterations (compared to 4-5 at $p = 0.05$), due to the reduced impact of random jumps. However, the increased iteration count has a minimal impact on its overall efficiency.

Effects of Increasing Network Size N from 100 to 500

As N increases from 100 to 500, individual PageRank values for top pages decrease due to the probability mass spreading across more nodes. Nonetheless, schema.org and social media pages remain top-ranked, as their extensive inbound links maintain their relative importance across all N sizes. For both methods, the power method's computational advantage over the eigenvector method becomes more pronounced with larger N, confirming its suitability for handling large networks efficiently.

In summary, increasing p from 0.05 to 0.10 shifts PageRank values slightly towards the network's true link structure, enhancing the prominence of highly connected pages. Both methods successfully capture this more refined distribution, though the power method remains more efficient and scales better with network size. This analysis shows how even a small increase in p heightens the sensitivity of PageRank to the network's connectivity, providing a clearer reflection of page importance based on link structure.

6. Results for Different Network Sizes (N) at $p = 0.85$

The top 10 pages and their probabilities for each N are presented below.

Eigenvector Method Results

N	Rank	Page (Probability)
N = 100		
	1	`https://schema.org` (0.431915)
	2	`https://www.facebook.com/TurnitinSoftware` (0.045361)
	3	`https://github.com/gradescope/autograder_samples/edit/master/docs/index.md` (0.020826)
	4	`https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion` (0.016046)
	5	`https://tv.youtube.com/learn/nflsundayticket` (0.013192)
	6	`https://www.linkedin.com/company/turnitin` (0.011833)
	7	`https://www.youtube.com/creators` (0.011833)
	8	`http://schema.org` (0.011542)
	9	`https://www.turnitin.com` (0.009461)
	10	`https://twitter.com/gradescope` (0.009247)
N = 200		
	1	`https://schema.org` (0.333654)
	2	`https://www.facebook.com/TurnitinSoftware` (0.040201)
	3	`https://www.mkdocs.org` (0.017457)
	4	`https://github.com/readthedocs/sphinx_rtd_theme` (0.017457)
	5	`https://github.com/gradescope/autograder_samples/edit/master/docs/index.md` (0.012627)
	6	`https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion` (0.011281)
	7	`https://www.linkedin.com/company/turnitin` (0.010487)
	8	`http://schema.org` (0.008649)
	9	`https://tv.youtube.com/learn/nflsundayticket` (0.008490)

N	Rank	Page (Probability)
	10	https://www.turnitin.com (0.008302)
N = 300		
	1	https://schema.org (0.111333)
	2	https://validator.schema.org (0.100964)
	3	https://www.facebook.com/TurnitinSoftware (0.027349)
	4	http://schema.org (0.013787)
	5	http://rdfs.org/ns/void# (0.012422)
	6	https://github.com/readthedocs/sphinx_rtd_theme (0.011868)
	7	https://www.mkdocs.org (0.011868)
	8	https://supportcenter.turnitin.com/s (0.010804)
	9	https://github.com/schemaorg/schemaorg/issues/2289 (0.009845)
	10	https://github.com/schemaorg/schemaorg/issues/2421 (0.009845)
N = 400		
	1	https://schema.org (0.095216)
	2	https://validator.schema.org (0.087305)
	3	http://community.forumbee.com/t/18hzyj/enabling-browser-cookies (0.014324)
	4	http://schema.org (0.012702)
	5	http://rdfs.org/ns/void# (0.011283)
	6	https://www.mkdocs.org (0.010098)
	7	https://github.com/readthedocs/sphinx_rtd_theme (0.010098)
	8	https://supportcenter.turnitin.com/s (0.009085)
	9	https://github.com/schemaorg/schemaorg/issues/2289 (0.008752)
	10	https://github.com/schemaorg/schemaorg/issues/2421 (0.008420)
N = 500		
	1	https://schema.org (0.052581)
	2	https://github.com/schemaorg/schemaorg (0.032448)
	3	https://validator.schema.org (0.024152)
	4	http://schema.org (0.014448)
	5	http://github.com/schemaorg/schemaorg (0.012845)
	6	http://blog.schema.org (0.012845)
	7	https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion (0.012384)
	8	http://community.forumbee.com/t/18hzyj/enabling-browser-cookies (0.012052)
	9	https://github.com/schemaorg/schemaorg/issues/2341 (0.009931)
	10	http://rdfs.org/ns/void# (0.009644)

Power Method Results

N	Rank	Page (Probability)
N = 100		
	1	https://schema.org (0.431914)
	2	https://www.facebook.com/TurnitinSoftware (0.045361)
	3	https://github.com/gradescope/autograder_samples/edit/master/docs/index.md (0.020826)
	4	https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion (0.016046)
	5	https://tv.youtube.com/learn/nflsundayticket (0.013192)
	6	https://www.linkedin.com/company/turnitin (0.011833)
	7	https://www.youtube.com/creators (0.011833)
	8	http://schema.org (0.011542)
	9	https://www.turnitin.com (0.009461)
	10	https://twitter.com/gradescope (0.009247)
N = 200		
	1	https://schema.org (0.333653)
	2	https://www.facebook.com/TurnitinSoftware (0.040200)

N	Rank	Page (Probability)
	3	`https://www.mkdocs.org` (0.017457)
	4	`https://github.com/readthedocs/sphinx_rtd_theme` (0.017457)
	5	`https://github.com/gradescope/autograder_samples/edit/master/docs/index.md` (0.012627)
	6	`https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion` (0.011281)
	7	`https://www.linkedin.com/company/turnitin` (0.010487)
	8	`http://schema.org` (0.008649)
	9	`https://tv.youtube.com/learn/nflsundayticket` (0.008490)
	10	`https://www.turnitin.com` (0.008302)
N = 300		
	1	`https://schema.org` (0.111333)
	2	`https://validator.schema.org` (0.100963)
	3	`https://www.facebook.com/TurnitinSoftware` (0.027349)
	4	`http://schema.org` (0.013787)
	5	`http://rdfs.org/ns/void#` (0.012422)
	6	`https://www.mkdocs.org` (0.011868)
	7	`https://github.com/readthedocs/sphinx_rtd_theme` (0.011868)
	8	`https://supportcenter.turnitin.com/s` (0.010804)
	9	`https://github.com/schemaorg/schemaorg/issues/2289` (0.009845)
	10	`https://github.com/schemaorg/schemaorg/issues/2421` (0.009845)
N = 400		
	1	`https://schema.org` (0.095215)
	2	`https://validator.schema.org` (0.087304)
	3	`http://community.forumbee.com/t/18hzyj/enabling-browser-cookies` (0.014324)
	4	`http://schema.org` (0.012702)
	5	`http://rdfs.org/ns/void#` (0.011283)
	6	`https://github.com/readthedocs/sphinx_rtd_theme` (0.010098)
	7	`https://www.mkdocs.org` (0.010098)
	8	`https://supportcenter.turnitin.com/s` (0.009085)
	9	`https://github.com/schemaorg/schemaorg/issues/2289` (0.008752)
	10	`https://github.com/schemaorg/schemaorg/issues/2421` (0.008420)
N = 500		
	1	`https://schema.org` (0.052582)
	2	`https://github.com/schemaorg/schemaorg` (0.032448)
	3	`https://validator.schema.org` (0.024152)
	4	`http://schema.org` (0.014448)
	5	`http://github.com/schemaorg/schemaorg` (0.012845)
	6	`http://blog.schema.org` (0.012845)
	7	`https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion` (0.012385)
	8	`http://community.forumbee.com/t/18hzyj/enabling-browser-cookies` (0.012052)
	9	`https://github.com/schemaorg/schemaorg/issues/2341` (0.009931)
	10	`http://rdfs.org/ns/void#` (0.009644)

Performance Metrics

N	Eigenvector Time (s)	Power Method Time (s)	Power Iterations
100	0.0029	0.0002	35
200	0.0076	0.0002	39
300	0.0230	0.0005	41
400	0.0377	0.0006	46
500	0.0583	0.0010	57

In-depth Analysis and Interpretation of Results for $p = 0.85$

Analysis of Results with High Damping Factor $p = 0.85$

At $p = 0.85$, both the eigenvector and power methods consistently converge to the stationary distribution across all network sizes N. The higher damping factor places greater emphasis on link-following, causing the PageRank algorithm to concentrate probability mass on well-connected pages. As N increases, individual PageRank values for top pages decrease due to the fixed probability mass distributed over more nodes, though this decrease is less pronounced than at lower p values. For instance, schema.org holds the top rank across all network sizes, capturing as much as 43.2 percent of the probability mass at $N = 100$. This increased concentration at $p = 0.85$ demonstrates how a higher damping factor enhances the influence of pages central to the network's structure.

Impact on Top-Ranked Pages

The emphasis on link-following allows pages such as schema.org and other technical resources—like validator.schema.org and GitHub documentation pages—to rise in prominence due to their significant inbound links. These pages benefit from the recursive nature of PageRank, which assigns higher importance to nodes receiving links from other influential pages. With only a 15 percent probability of random jumps, the algorithm prioritizes pages deeply integrated within the network, resulting in higher PageRank values for these technical and frequently cited sites. This effect highlights the impact of link structure when random jumps are less frequent.

Computational Efficiency: Eigenvector vs. Power Method with High p

The higher $p = 0.85$ value affects the computational efficiency of the power method, requiring up to 57 iterations at $N = 500$, as opposed to the 4-6 iterations typical at lower p values. This is due to the transition matrix:

$$P = 0.85 \times P_{\text{original}} + 0.15 \times \frac{1}{N} \times \mathbf{E}$$

where the dominant link-following component $0.85 \times P_{\text{original}}$ results in a smaller spectral gap. This narrower gap slows the convergence of the power method by reducing the rate at which the probability mass stabilizes. However, the eigenvector method remains unaffected in terms of iteration count, with a consistent $O(N^3)$ complexity due to its reliance on eigenvalue decomposition. Even though the number of iterations of the power method increased significantly, it's still much faster than the eigenvector method and that's simply because $O(N^3)$ grows much faster than $O(N * iterations)$. But I suspect we might see these two numbers begin to converge more as the damping factor approaches 1.

Effects of Increasing Network Size N from 100 to 500

As N increases from 100 to 500, individual PageRank values for top-ranked pages decrease, as the fixed probability mass is spread over more pages. Despite this, schema.org and social media pages consistently rank highly, as their extensive inbound links maintain their relative importance. The power method's efficiency advantage over the eigenvector method becomes more noticeable with larger N, as the power method leverages sparse structures for quicker computations, while the eigenvector method's $O(N^3)$ complexity grows more rapidly.

In summary, with $p = 0.85$, the PageRank distribution more closely reflects the network's connectivity structure, assigning higher values to well-connected nodes. Both methods successfully capture this distribution, though the power method's iteration count increases due to slower convergence. This analysis shows how a higher p amplifies the influence of network topology, enhancing the representation of page importance based on connectivity while maintaining efficiency, particularly in the power method.

7. Results for Different Network Sizes (N) at $p = 0.90$

The top 10 pages and their probabilities for each N are presented below.

Eigenvector Method Results

N	Rank	Page (Probability)
N = 100		
	1	https://schema.org (0.537316)
	2	https://www.facebook.com/TurnitinSoftware (0.053325)
	3	https://github.com/gradescope/autograder_samples/edit/master/docs/index.md (0.022820)
	4	https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion (0.012891)
	5	https://tv.youtube.com/learn/nflsundayticket (0.010980)
	6	https://www.linkedin.com/company/turnitin (0.009695)
	7	https://www.youtube.com/creators (0.009617)
	8	http://schema.org (0.009560)
	9	https://www.turnitin.com (0.007514)
	10	https://twitter.com/gradescope (0.007081)
N = 200		
	1	https://schema.org (0.436995)
	2	https://www.facebook.com/TurnitinSoftware (0.050013)
	3	https://www.mkdocs.org (0.020217)
	4	https://github.com/readthedocs/sphinx_rtd_theme (0.020217)
	5	https://github.com/gradescope/autograder_samples/edit/master/docs/index.md (0.014390)
	6	https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion (0.009475)
	7	https://www.linkedin.com/company/turnitin (0.009093)
	8	http://schema.org (0.007556)
	9	https://tv.youtube.com/learn/nflsundayticket (0.007320)
	10	https://www.turnitin.com (0.006975)
N = 300		
	1	https://schema.org (0.145484)
	2	https://validator.schema.org (0.136350)
	3	https://www.facebook.com/TurnitinSoftware (0.034310)
	4	http://rdfs.org/ns/void# (0.014917)
	5	https://www.mkdocs.org (0.013933)
	6	https://github.com/readthedocs/sphinx_rtd_theme (0.013933)
	7	http://schema.org (0.012012)
	8	https://github.com/schemaorg/schemaorg/issues/2421 (0.011557)
	9	https://github.com/schemaorg/schemaorg/issues/2289 (0.011557)
	10	https://github.com/schemaorg/schemaorg/issues/2454 (0.011046)
N = 400		
	1	https://schema.org (0.128648)
	2	https://validator.schema.org (0.121435)
	3	http://community.forumbee.com/t/18hzyj/enabling-browser-cookies (0.017654)
	4	http://rdfs.org/ns/void# (0.014077)
	5	https://github.com/readthedocs/sphinx_rtd_theme (0.012295)
	6	https://www.mkdocs.org (0.012295)
	7	http://schema.org (0.011326)
	8	https://github.com/schemaorg/schemaorg/issues/2289 (0.010687)
	9	https://github.com/schemaorg/schemaorg/issues/2421 (0.010257)
	10	https://github.com/schemaorg/schemaorg/issues/2927 (0.010235)
N = 500		
	1	https://schema.org (0.062055)

N	Rank	Page (Probability)
	2	https://github.com/schemaorg/schemaorg (0.039882)
	3	https://validator.schema.org (0.028866)
	4	http://community.forumbee.com/t/18hzyj/enabling-browser-cookies (0.015709)
	5	http://github.com/schemaorg/schemaorg (0.014904)
	6	http://blog.schema.org (0.014904)
	7	http://schema.org (0.013532)
	8	https://github.com/schemaorg/schemaorg/issues/2341 (0.012967)
	9	http://rdfs.org/ns/void# (0.012732)
	10	https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion (0.011391)

Power Method Results

N	Rank	Page (Probability)
N = 100		
	1	https://schema.org (0.537314)
	2	https://www.facebook.com/TurnitinSoftware (0.053325)
	3	https://github.com/gradescope/autograder_samples/edit/master/docs/index.md (0.022820)
	4	https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion (0.012892)
	5	https://tv.youtube.com/learn/nflsundayticket (0.010980)
	6	https://www.linkedin.com/company/turnitin (0.009696)
	7	https://www.youtube.com/creators (0.009618)
	8	http://schema.org (0.009560)
	9	https://www.turnitin.com (0.007514)
	10	https://twitter.com/gradescope (0.007081)
N = 200		
	1	https://schema.org (0.436993)
	2	https://www.facebook.com/TurnitinSoftware (0.050013)
	3	https://github.com/readthedocs/sphinx_rtd_theme (0.020217)
	4	https://www.mkdocs.org (0.020217)
	5	https://github.com/gradescope/autograder_samples/edit/master/docs/index.md (0.014390)
	6	https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion (0.009476)
	7	https://www.linkedin.com/company/turnitin (0.009093)
	8	http://schema.org (0.007556)
	9	https://tv.youtube.com/learn/nflsundayticket (0.007320)
	10	https://www.turnitin.com (0.006975)
N = 300		
	1	https://schema.org (0.145483)
	2	https://validator.schema.org (0.136350)
	3	https://www.facebook.com/TurnitinSoftware (0.034310)
	4	http://rdfs.org/ns/void# (0.014917)
	5	https://www.mkdocs.org (0.013933)
	6	https://github.com/readthedocs/sphinx_rtd_theme (0.013933)
	7	http://schema.org (0.012012)
	8	https://github.com/schemaorg/schemaorg/issues/2421 (0.011556)
	9	https://github.com/schemaorg/schemaorg/issues/2289 (0.011556)
	10	https://github.com/schemaorg/schemaorg/issues/2454 (0.011046)
N = 400		
	1	https://schema.org (0.128647)
	2	https://validator.schema.org (0.121434)
	3	http://community.forumbee.com/t/18hzyj/enabling-browser-cookies (0.017654)
	4	http://rdfs.org/ns/void# (0.014077)
	5	https://github.com/readthedocs/sphinx_rtd_theme (0.012295)

N	Rank	Page (Probability)
	6	https://www.mkdocs.org (0.012295)
	7	http://schema.org (0.011326)
	8	https://github.com/schemaorg/schemaorg/issues/2289 (0.010687)
	9	https://github.com/schemaorg/schemaorg/issues/2421 (0.010257)
	10	https://github.com/schemaorg/schemaorg/issues/2927 (0.010235)
N = 500		
	1	https://schema.org (0.062055)
	2	https://github.com/schemaorg/schemaorg (0.039882)
	3	https://validator.schema.org (0.028866)
	4	http://community.forumbee.com/t/18hzyj/enabling-browser-cookies (0.015709)
	5	http://github.com/schemaorg/schemaorg (0.014904)
	6	http://blog.schema.org (0.014904)
	7	http://schema.org (0.013532)
	8	https://github.com/schemaorg/schemaorg/issues/2341 (0.012967)
	9	http://rdfs.org/ns/void# (0.012732)
	10	https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion (0.011391)

Performance Metrics

N	Eigenvector Time (s)	Power Method Time (s)	Power Iterations
100	0.0028	0.0002	40
200	0.0093	0.0002	47
300	0.0222	0.0005	51
400	0.0396	0.0017	59
500	0.0541	0.0010	57

Analysis of Results with Very High Damping Factor $p = 0.90$

With a damping factor of $p = 0.90$, both the eigenvector and power methods converge consistently across all network sizes N, producing stable PageRank values that emphasize link-following. The high p value intensifies the algorithm's dependence on network structure, causing well-connected pages to hold a larger share of the probability mass. As N increases, individual PageRank values for top pages decrease as the probability mass spreads across more nodes, though this effect is less pronounced at $p = 0.90$ than at lower values. For instance, schema.org captures 53.73% of the probability mass at $N = 100$, underscoring the increased importance of highly connected pages as p approaches 1.

Impact on Top-Ranked Pages

The high emphasis on link-following allows pages like schema.org, validator.schema.org, and GitHub documentation pages to gain prominence due to their substantial inbound links from other important pages. These pages benefit from the recursive nature of PageRank, where links from other high-ranking nodes reinforce their influence. With a random jump probability of only 10 percent, peripheral pages have less opportunity to gain PageRank, while highly connected nodes increasingly dominate the distribution. This setting highlights the reinforcing effect on the importance of central, highly linked pages.

Computational Efficiency: Eigenvector vs. Power Method with Very High p

At $p = 0.90$, the power method requires significantly more iterations to converge—up to 59 iterations for $N = 400$—compared to lower values like $p = 0.85$, which required around 35-57 iterations. This increase in iteration count results from the transition matrix structure:

$$P = 0.90 \times P_{\text{original}} + 0.10 \times \frac{1}{N} \times \mathbf{E}$$

With $0.90 \times P_{\text{original}}$ dominating, the spectral gap between the largest and second-largest eigenvalues narrows, slowing convergence. This effect requires additional iterations in the power method to reach steady-state, while the eigenvector method's $O(N^3)$ complexity remains unaffected by changes in p and depends solely on network size N. But the increase is still not enough to truly affect computational efficiency.

Effects of Increasing Network Size N from 100 to 500

As network size N increases from 100 to 500, individual PageRank values for top pages diminish due to the fixed probability mass spread across more nodes. Nonetheless, schema.org and other highly linked pages retain their high ranks as their connectivity sustains their relative importance across network sizes. The power method's advantage in computational efficiency over the eigenvector method becomes increasingly clear at larger N, as the sparse matrix structure of P allows it to handle larger networks more efficiently, despite the additional iterations required for high p values.

In summary, at $p = 0.90$, the PageRank distribution is more concentrated on well-connected pages, with both methods accurately capturing this distribution. However, the power method requires more iterations to converge as p increases, highlighting the trade-off between convergence speed and network topology sensitivity. This analysis shows that the power method remains practical and efficient for large networks, while a higher p enhances the algorithm's emphasis on connectivity, making PageRank values more reflective of network structure.

8. Results for Different Network Sizes (N) at $p = 0.95$

The top 10 pages and their probabilities for each N are presented below.

Eigenvector Method Results

N	Rank	Page (Probability)
N = 100		
	1	`https://schema.org` (0.682465)
	2	`https://www.facebook.com/TurnitinSoftware` (0.064070)
	3	`https://github.com/gradescope/autograder_samples/edit/master/docs/index.md` (0.025397)
	4	`https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion` (0.007901)
	5	`https://tv.youtube.com/learn/nflsundayticket` (0.007027)
	6	`https://www.linkedin.com/company/turnitin` (0.006102)
	7	`http://schema.org` (0.006085)
	8	`https://www.youtube.com/creators` (0.005996)
	9	`https://www.turnitin.com` (0.004572)
	10	`https://twitter.com/gradescope` (0.004127)
N = 200		
	1	`https://schema.org` (0.590270)
	2	`https://www.facebook.com/TurnitinSoftware` (0.064219)
	3	`https://www.mkdocs.org` (0.024046)
	4	`https://github.com/readthedocs/sphinx_rtd_theme` (0.024046)
	5	`https://github.com/gradescope/autograder_samples/edit/master/docs/index.md` (0.016845)
	6	`https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion` (0.006137)
	7	`https://www.linkedin.com/company/turnitin` (0.006116)
	8	`http://schema.org` (0.005127)
	9	`https://tv.youtube.com/learn/nflsundayticket` (0.004898)
	10	`https://www.turnitin.com` (0.004534)
N = 300		
	1	`https://schema.org` (0.197693)
	2	`https://validator.schema.org` (0.191425)

N	Rank	Page (Probability)
	3	https://www.facebook.com/TurnitinSoftware (0.044736)
	4	http://rdfs.org/ns/void# (0.018627)
	5	https://github.com/readthedocs/sphinx_rtd_theme (0.016941)
	6	https://www.mkdocs.org (0.016941)
	7	https://github.com/schemaorg/schemaorg/issues/2289 (0.014084)
	8	https://github.com/schemaorg/schemaorg/issues/2421 (0.014084)
	9	https://github.com/schemaorg/schemaorg/issues/2927 (0.013423)
	10	https://github.com/schemaorg/schemaorg/issues/3392 (0.013423)
N = 400		
	1	https://schema.org (0.184124)
	2	https://validator.schema.org (0.178898)
	3	http://community.forumbee.com/t/18hzyj/enabling-browser-cookies (0.022955)
	4	http://rdfs.org/ns/void# (0.018614)
	5	https://www.mkdocs.org (0.015808)
	6	https://github.com/readthedocs/sphinx_rtd_theme (0.015808)
	7	https://github.com/schemaorg/schemaorg/issues/2289 (0.013815)
	8	https://github.com/schemaorg/schemaorg/issues/2421 (0.013227)
	9	https://github.com/schemaorg/schemaorg/issues/2927 (0.013196)
	10	https://en.wikipedia.org/wiki/JSON-LD (0.013097)
N = 500		
	1	https://schema.org (0.068087)
	2	https://github.com/schemaorg/schemaorg (0.045515)
	3	https://validator.schema.org (0.032017)
	4	http://community.forumbee.com/t/18hzyj/enabling-browser-cookies (0.023527)
	5	https://github.com/schemaorg/schemaorg/issues/2341 (0.019514)
	6	http://rdfs.org/ns/void# (0.019401)
	7	https://www.mkdocs.org (0.016188)
	8	https://github.com/readthedocs/sphinx_rtd_theme (0.016188)
	9	http://github.com/schemaorg/schemaorg (0.016025)
	10	http://blog.schema.org (0.016025)

Power Method Results

N	Rank	Page (Probability)
N = 100		
	1	https://schema.org (0.682463)
	2	https://www.facebook.com/TurnitinSoftware (0.064070)
	3	https://github.com/gradescope/autograder_samples/edit/master/docs/index.md (0.025397)
	4	https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion (0.007901)
	5	https://tv.youtube.com/learn/nflsundayticket (0.007027)
	6	https://www.linkedin.com/company/turnitin (0.006102)
	7	http://schema.org (0.006085)
	8	https://www.youtube.com/creators (0.005996)
	9	https://www.turnitin.com (0.004572)
	10	https://twitter.com/gradescope (0.004127)
N = 200		
	1	https://schema.org (0.590268)
	2	https://www.facebook.com/TurnitinSoftware (0.064219)
	3	https://www.mkdocs.org (0.024046)
	4	https://github.com/readthedocs/sphinx_rtd_theme (0.024046)
	5	https://github.com/gradescope/autograder_samples/edit/master/docs/index.md (0.016845)
	6	https://twitter3e4tixl4xyajtrzo62zg5vztmjuricljdp2c5kshju4avyoid.onion (0.006137)

N	Rank	Page (Probability)
	7	`https://www.linkedin.com/company/turnitin` (0.006116)
	8	`http://schema.org` (0.005127)
	9	`https://tv.youtube.com/learn/nflsundayticket` (0.004899)
	10	`https://www.turnitin.com` (0.004534)
N = 300		
	1	`https://schema.org` (0.197692)
	2	`https://validator.schema.org` (0.191424)
	3	`https://www.facebook.com/TurnitinSoftware` (0.044735)
	4	`http://rdfs.org/ns/void#` (0.018627)
	5	`https://www.mkdocs.org` (0.016941)
	6	`https://github.com/readthedocs/sphinx_rtd_theme` (0.016941)
	7	`https://github.com/schemaorg/schemaorg/issues/2289` (0.014084)
	8	`https://github.com/schemaorg/schemaorg/issues/2421` (0.014084)
	9	`https://github.com/schemaorg/schemaorg/issues/2927` (0.013423)
	10	`https://github.com/schemaorg/schemaorg/issues/3392` (0.013423)
N = 400		
	1	`https://schema.org` (0.184123)
	2	`https://validator.schema.org` (0.178897)
	3	`http://community.forumbee.com/t/18hzyj/enabling-browser-cookies` (0.022955)
	4	`http://rdfs.org/ns/void#` (0.018614)
	5	`https://www.mkdocs.org` (0.015808)
	6	`https://github.com/readthedocs/sphinx_rtd_theme` (0.015808)
	7	`https://github.com/schemaorg/schemaorg/issues/2289` (0.013815)
	8	`https://github.com/schemaorg/schemaorg/issues/2421` (0.013227)
	9	`https://github.com/schemaorg/schemaorg/issues/2927` (0.013196)
	10	`https://en.wikipedia.org/wiki/JSON-LD` (0.013097)
N = 500		
	1	`https://schema.org` (0.068089)
	2	`https://github.com/schemaorg/schemaorg` (0.045515)
	3	`https://validator.schema.org` (0.032017)
	4	`http://community.forumbee.com/t/18hzyj/enabling-browser-cookies` (0.023527)
	5	`https://github.com/schemaorg/schemaorg/issues/2341` (0.019513)
	6	`http://rdfs.org/ns/void#` (0.019401)
	7	`https://github.com/readthedocs/sphinx_rtd_theme` (0.016188)
	8	`https://www.mkdocs.org` (0.016188)
	9	`http://github.com/schemaorg/schemaorg` (0.016026)
	10	`http://blog.schema.org` (0.016026)

Performance Metrics

N	Eigenvector Time (s)	Power Method Time (s)	Power Iterations
100	0.0038	0.0005	48
200	0.0086	0.0004	57
300	0.0176	0.0005	64
400	0.0372	0.0010	79
500	0.0494	0.0022	144

Analysis of Results with Extremely High Damping Factor $p = 0.95$

At $p = 0.95$, both the eigenvector and power methods converge consistently across network sizes N, indicating stable results. The high p value emphasizes link-following over random jumps, concentrating PageRank values among the most connected pages. As N increases, the individual PageRank values decrease as probability mass is shared across more nodes, but top-ranked pages still retain a substantial portion of the probability mass. For instance, schema.org holds roughly 68.25% of the probability mass at $N = 100$, reflecting the intense focus on well-connected pages when the algorithm relies predominantly on link-following.

Impact on Top-Ranked Pages

At $p = 0.95$, link-following dominance makes schema.org and other highly connected pages, like validator.schema.org and GitHub resources, the primary recipients of PageRank. These pages benefit from recursive importance, where links from other high-ranking pages reinforce their ranks. With only a 5 percent chance of random jumps, lesser-connected pages have minimal opportunities to accrue PageRank, leading to a highly concentrated distribution that highlights central nodes with extensive inbound links.

Computational Efficiency: Eigenvector vs. Power Method with High p

The high p value significantly impacts the power method's convergence rate, requiring up to 144 iterations at $N = 500$, compared to much lower iteration counts at smaller p values. This increase is due to the dominance of the link-following term in the transition matrix:

$$P = 0.95 \times P_{\text{original}} + 0.05 \times \frac{1}{N} \times \mathbf{E}$$

where the heavy weight on P_{original} reduces the spectral gap, resulting in slower convergence as the probability mass redistributes less effectively across iterations. In contrast, the eigenvector method remains unaffected by changes in p, maintaining a stable $O(N^3)$ complexity based solely on N. But even after a massive increase in computation time, power method is still the more efficient method as its still significantly quicker.

Effects of Increasing Network Size N from 100 to 500

As N scales from 100 to 500, individual PageRank scores for top pages decrease as the fixed probability mass distributes over a larger network. Nonetheless, schema.org and other highly linked pages consistently rank at the top, as their high connectivity sustains their relative importance across all network sizes. The power method's efficiency advantage over the eigenvector method becomes more evident with larger N, despite the higher iteration counts needed at $p = 0.95$, as its sparse operations enable better scalability.

In summary, at $p = 0.95$, the PageRank algorithm produces a distribution concentrated on well-connected pages, accurately captured by both methods. However, the power method requires substantially more iterations to converge due to the smaller spectral gap, illustrating the trade-off between convergence speed and the emphasis on network structure. This analysis shows that while the power method remains effective for large networks, high p values heighten the algorithm's sensitivity to link structure, favoring central pages with high connectivity.

9. Final Discussion: General Trends, Consistencies, and Insights on PageRank Behavior Across Damping Factors and Network Sizes

The results from varying damping factors p and network sizes N reveal distinct trends and insights into how PageRank behaves and adapts to changes in network structure and algorithmic settings. In this section, I will synthesize these findings to highlight the key patterns observed, focusing on page consistency, damping factor influence, computational efficiency, and the implications of scaling up network size.

Consistent Top-Ranked Pages Across Damping Factors and Network Sizes

A striking consistency throughout all damping factor settings is the prominence of highly connected hubs, notably `schema.org`. Regardless of p or N, `schema.org` emerges as the top-ranked page, frequently holding a substantial portion of the probability mass, especially at higher damping values. Other websites such as `validator.schema.org` and GitHub documentation pages also consistently achieve high ranks due to their dense linkage, reinforcing their central positions in the network.

The resilience of these pages across different settings indicates a robust structural advantage—they are key nodes in the network's topology with extensive inbound links, making them influential within the network's connective architecture. This structural advantage is further amplified as p increases because a higher p value means the PageRank algorithm relies more heavily on the link-following behavior, reinforcing the importance of well-connected pages. With each incremental increase in p, these high-connectivity pages receive a larger share of the probability mass, as the random jump probability diminishes and the algorithm's behavior increasingly mirrors the natural link structure of the network, demonstrating how the PageRank algorithm prioritizes link-following, effectively locking in the prominence of these well-connected nodes.

Impact of Damping Factor p: From Uniform Distribution to Extreme Concentration

The damping factor p plays a crucial role in shaping how PageRank distributes probability mass across a network, ranging from a more uniform spread at low p values to a highly concentrated distribution at high p values. As p approaches 0, the PageRank algorithm increasingly emphasizes random jumps, leading to an almost even probability distribution across all pages, effectively minimizing the influence of the network's link structure. This behavior reduces the differentiation between nodes, even for well-connected hubs, as random jumps dominate the algorithm's process. Conversely, as p approaches 1, PageRank relies more heavily on link-following, causing the probability mass to concentrate sharply among the most connected nodes, with these "hub" pages capturing a disproportionately high share of importance. For instance, at a high p value such as 0.95, central pages like `schema.org` can capture over 68% of the total probability mass, reflecting an extreme level of centrality and influence. If we were to push these values to more extreme cases, approaching $p = 0$ or $p = 1$, the network would respectively show either a nearly flat distribution, ignoring connectivity entirely, or an absolute hierarchy dominated by only the most linked pages. This range of behaviors illustrates how p modulates PageRank from a "random surfer" model at low values to a "network structure detector" at high values, underscoring its sensitivity to the damping parameter.

Computational Efficiency: Eigenvector vs. Power Method

The power method consistently demonstrates greater computational efficiency than the eigenvector approach across all tested damping factors and network sizes, even though it requires a higher number of iterations at elevated p values. For low to moderate p values, the power method converges swiftly, often within just 4 to 6 iterations. This efficiency is largely due to a broader spectral gap, which enables the probability mass to redistribute and stabilize rapidly. In contrast, the eigenvector method, with its $O(N^3)$ complexity, faces exponentially increasing computational demands as the network size N grows.

As p approaches higher values (e.g., $p = 0.95$), the power method requires significantly more iterations to converge, up to 144 for $N = 500$, because the spectral gap narrows, slowing down redistribution. Nevertheless, the power method's $O(N^2)$ complexity per iteration, resulting from the fully dense transition matrix, still provides a computational advantage over the eigenvector approach, which is hindered by its cubic complexity. If p were to approach even closer to 1, the power method would require more iterations for convergence, but its iterative nature would still make it more scalable than the eigenvector method, especially for large N, despite the loss of sparsity in the transition matrix.

Tolerance of the power method

The tolerance of the power method, which sets the threshold for convergence, has a significant impact on the number of iterations, computational efficiency, and result accuracy. When we set a tighter tolerance (e.g., 10^{-9} instead of 10^{-6}), the power method requires more iterations to reach the target precision, as the difference between consecutive rank vectors must be smaller before it stops. This leads to higher computational costs but produces a more accurate PageRank vector that closely approximates the steady-state distribution. Conversely, a looser tolerance (e.g., 10^{-4}) allows the algorithm to converge more quickly with fewer iterations, improving efficiency at the expense of a slight reduction in accuracy. However, the trade-off is often acceptable in practical applications, as small deviations in the rank vector might not meaningfully impact the relative ranking of nodes.

In PageRank computations, a stricter tolerance may be necessary under certain conditions to ensure accurate results. As network size N increases, the probability mass is spread more thinly across nodes, leading to smaller PageRank values for each, and a tighter tolerance can help capture these finer differences, especially in large networks where accurate rankings of many nodes are important. Additionally, applications requiring high precision, such as financial modeling or competitive ranking, benefit from stricter tolerance to minimize approximation error. Similarly, sparse or highly asymmetric networks, where a few nodes have many links while others have few or none, may need stricter tolerance to ensure that PageRank values reflect these structural nuances. However, in our experiment, the current tolerance level is appropriate because the results closely align with the eigenvector method's outputs. Since the eigenvector method provides the exact steady-state distribution, any similar results obtained by the power method suggest that the tolerance level is sufficiently tight to produce an accurate approximation, with the trade off of much more efficient computation. In this case as long as the matrix structure does not change too drastically and the size of the network does not increase too significantly its safe to use the power method with out existing tolerance.

However, to determine the ideal balance without relying on the eigenvector method—which could be beneficial for maximizing computational efficiency—one can assess the sensitivity of the PageRank rankings to different tolerance levels. If small changes in tolerance do not significantly alter the top-ranking nodes or the overall distribution, a looser tolerance might be sufficient, providing faster results with minimal accuracy loss.

Scaling Up Network Size N: Predictions for Larger Networks

As network size N increases from 100 to 500 and beyond, individual PageRank values for top-ranked pages decrease as the fixed probability mass spreads across a growing number of nodes. This trend would likely persist with even larger networks (e.g., $N = 1000$ or $N = 10000$), where the most connected pages would continue to dominate in rank relative to others, though with reduced absolute values. Structurally central pages, like `schema.org`, consistently retain high ranks across network sizes due to their dense inbound linkages, suggesting that in larger networks, these core pages would still lead, albeit with a smaller share of the total probability mass due to broader distribution.

With very large network sizes, the results produced by the power method and the eigenvector method could diverge significantly due to inherent differences in their computational approaches and the handling of high-dimensional matrices. The power method is especially efficient for large, sparse networks, as it iteratively updates the probability vector based only on non-zero entries in the adjacency matrix. This is effective for networks with many nodes but relatively few connections per node, allowing the power method to scale well even when N is extremely large, as it only processes non-zero links in each iteration.

In contrast, the eigenvector method requires calculating the entire matrix's eigenvalues and eigenvectors, which becomes computationally infeasible for very large N due to its $O(N^3)$ complexity. It also demands that the entire adjacency matrix be stored in memory, which is impractical for large, sparse networks. Additionally, the eigenvector method's sensitivity to rounding errors and precision limitations is exacerbated with high-dimensional matrices, where such errors can accumulate, potentially leading to inaccuracies in the ranking order of pages. Thus, for very large networks, the power method is likely to provide faster, more scalable results with stable rankings of top nodes and minimal computational overhead, whereas the eigenvector method might suffer from performance inefficiencies and rounding errors that could distort rankings, especially for nodes with similar centrality scores.

The computational demand of the eigenvector method increases significantly with network size, making it less practical for large networks. The power method, with $O(N^2 \times \text{iterations})$ complexity due to the fully dense transition matrix, remains feasible if damping values are moderate, thus limiting iteration counts. However, for high p values, the hundreds of iterations required for convergence may still present a challenge as the network size scales upward. Achieving convergence at high p values becomes difficult because PageRank values shift more gradually, leading to slower stabilization of the probability mass, especially as network size grows. This slow convergence stems from a smaller spectral gap, which causes the power method to require more iterations to settle, even with its iterative nature.

10. Alternative Model

To enhance the traditional PageRank algorithm, we propose the "Intelligent Surfer" model, which incorporates topic relevance to produce more meaningful and customized page rankings. This model modifies the random jump component by introducing a personalization vector that reflects the relevance of each page to a specific topic.

To create this personalization vector, we first define the topic of interest and analyze each page's content. For each page, we calculate a raw relevance score based on the presence of topic-specific keywords in titles, headings, body content, and, to a lesser extent, URL keywords. Each section is assigned a weight based on its importance, and the relevance score r_i for page i can be represented as a weighted sum:

$$r_i = w_{\text{title}} \cdot k_{\text{title}} + w_{\text{heading}} \cdot k_{\text{heading}} + w_{\text{body}} \cdot k_{\text{body}} + w_{\text{url}} \cdot k_{\text{url}}$$

where w represents weights for each section, and k represents the count of topic-specific keywords.

Next, we normalize these scores to form the personalization vector, ensuring the probabilities across pages sum to 1. For each page i, the normalized score p_i is given by:

$$p_i = \frac{r_i}{\sum_j r_j}$$

This personalization vector $\vec{p}$ thus reflects the probability of jumping to each page based on topic relevance.

To implement this, we integrate the personalization vector into the PageRank algorithm by modifying the random jump component. Here's how the Intelligent Surfer model transition probability matrix can be formulated:

$$P = p \times P_{\text{original}} + (1 - p) \times \mathbf{p}^T$$

where:

- p is the damping factor (similar to the original).
- P_{original} is the original link-based transition matrix.
- $\mathbf{p}$ is the topic-sensitive personalization vector, with each entry p_i representing the probability of jumping to page i based on its relevance to the topic (as determined by content analysis).
- The vector $\mathbf{p}$ replaces the uniform jump probability $\frac{1}{N}$ in the original model, creating a more customized and topic-relevant transition matrix.

This Intelligent Surfer model improves on the standard random surfer model by providing a more user-centric ranking. It mitigates the tendency of traditional PageRank to favor highly connected pages that may not be relevant to the topic, instead allowing important but less-connected pages to gain appropriate prominence.

For example, in an academic research network, a researcher interested in machine learning could use this model to prioritize papers related to that field. By creating a personalization vector with keywords like "neural networks," "deep learning," and "algorithm," and considering relevant terms in URLs, the modified PageRank algorithm would rank machine learning papers higher, allowing the researcher to quickly identify influential publications within the topic, even if they aren't the most highly linked.

In this context, the Intelligent Surfer model offers a more nuanced and practical approach to PageRank, making it highly effective for specialized applications and targeted content discovery.

11. Conclusion

The overall trends across different damping factors and network sizes emphasize the dual nature of PageRank as both a random surfer and network structure detector. Low p values highlight randomness, creating a nearly uniform distribution, while high p values amplify connectivity, placing heavy importance on well-linked pages.

The power method stands out as the more scalable approach for larger networks, but its efficiency can be hampered at extremely high p values due to slowed convergence. Nevertheless, it is still more favorable, especially with larger networks.

With larger networks, we can expect highly connected pages to retain their ranks but with a proportionally lower share of probability mass, as mass distribution becomes more diluted. These insights underscore PageRank's utility in analyzing connectivity but also reveal the algorithm's sensitivity to parameter choices, which can significantly impact computational feasibility and result interpretation.