Markov Chain and its application in the Google PageRank Algorithm

Hoàng Lê Anh Khoa
Faculty of Information Science and Engineering
University of Information Technology
Ho Chi Minh City, Vietnam
22520667@gm.uit.edu.vn

Ngô Nhất Khánh
Faculty of Information Science and Engineering
University of Information Technology
Ho Chi Minh City, Vietnam
22520640@gm.uit.edu.vn

Lê Vĩnh Kỳ
Faculty of Information Science and Engineering
University of Information Technology
Ho Chi Minh City, Vietnam
22520730@gm.uit.edu.vn

***Abstract*— In an era where the fourth industrial revolution is taking place, the link analysis algorithms for web search engines determine the importance and relevance of websites. Among the link analysis algorithms, PageRank is the modern ranking mechanism used in today's Google search engines. The PageRank algorithm is modeled as the behavior of a random surfer; this model can be seen as a Markov sequence to predict the actions of a system going from one state to another, taking into account only the current state. However, this model has problems with hanging nodes because these nodes cannot be presented in the Markov string model. This paper focuses on applying the Markov string to the PageRank algorithm and discusses a few methods to deal with nodal inclination.**

Keywords — Markov Chain; Web Graph; Random Walk; PageRank; Scaled PageRank; Transition Matrix; Dangling Page; Damping Factor.

I. INTRODUCTION (*HEADING 1*)

The Google PageRank algorithm stands as a pioneering application of Markov Chain theory in the realm of web search and information retrieval. Conceived by Larry Page and Sergey Brin during their time at Stanford University, PageRank fundamentally transformed how search engines assess and rank web pages. At its core, PageRank leverages the concept of a Markov Chain, treating web pages as states and hyperlinks as transitions between these states. This innovative approach introduced a paradigm shift, emphasizing the importance of link quality over sheer quantity. By iteratively traversing this network of links, PageRank assigns numerical scores to pages, reflecting their relative significance within the web's vast interconnectedness. This algorithm not only revolutionized Google's early search engine but also laid the groundwork for modern search methodologies. Today, PageRank, along with other sophisticated algorithms and machine learning techniques, continues to play a pivotal role in delivering accurate and relevant search results to users worldwide. Understanding PageRank's foundation in Markov Chain Theory provides valuable insight into its underlying principles and its enduring impact on the landscape of online information retrieval.

In this paper, we focus on the Markov chain and its application, which is the Google PageRank algorithm. In Section I, we introduce PageRank, which is a query, and content-independent algorithm. This is an application of the Markov chain.

Section II presents the theory of Markov chains and its practical applications in a few real-time scenarios.

Section III contains an explanation of the PageRank algorithm. The method known as PageRank simply displays the Web sites that match the query word to the user when they input it into a search engine, arranged according to their PageRank. The mathematical model behind it is incredible, even if it appears simple.

The application of the Markov chain is presented in Part IV. Implementing PageRank on a famous social network. In this section, our team has provided input data, output data, and tools to process and solve these problems.

II. MARKOV CHAIN

A. *Introduction to Markov Chain*

A stochastic process is a mathematical model for the random evolution of a system over time. More precisely, it is a collection of random variables $X_t(\omega)$ on a probability space Ω , indexed by a time parameter $t \in I$ from some index set I, and taking values in a common state space S.

Let $(\Omega, \mathrm{F}, \mathrm{P})$ be the probability space as defined above and $(X_k)_{0\le k\le n}$ the random vector whose components are the coordinate random variables as defined above. Then, for all $0 \le i \le n-1$ and for all $x_0, x_1, ..., x_{i+1} \in S,$ consider time i as representing the presence. Then, depending on the event $\{X_i = x_i\}$, the past and future of the process are independent. That is, for all $n > i$ and for all $x_0, x_1, ..., x_{i+1}, x_{i+1}, x_n \in S$,

$$P\left(X_{i+1} = x_{i+1} \mid X_0 = x_0, ..., X_i = x_i\right) = P\left(X_{i+1} = x_{i+1} \mid X_i = x_i\right) = p(x_i, x_{i+1})$$

The fundamental assumption on stochastic processes $(X_n)_{n\ge 0}$ is the so-called Markov property.

Let S be a (finite or countably infinite) discrete set. A discrete-time stochastic process $(X_n)_{n\ge 0}$ with state space S is called a **Markov chain** if for all $n \ge 0$ and all states $x_0, ..., x_{n-1}, x, y \in S,$

$$P\left(X_{n+1}=y \mid X_0=x_0,\ldots,X_{n-1}=x_{n-1},X_n=x\right)$$
$$=P\left(X_{n+1}=y \mid X_n=x\right)$$

B. Transition Matrix

Since we will always work with a discrete state space S, we can take $S=\{0,1,\ldots,N\}$, or in the infinite case, $S=N_0$. Using the natural ordering of the elements in *S*, it will often be convenient to write the transition probabilities Pxy in matrix format. This results in the (finite or infinite) **one-step transition matrix**, or simply transition matrix.

$$P=\begin{pmatrix} P_{00} & P_{01} & \cdots & P_{0N} \\ P_{10} & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ P_{N0} & \cdots & \cdots & P_{NN} \end{pmatrix} \text{ or } P=\begin{pmatrix} P_{00} & P_{01} & P_{02} & \cdots \\ P_{10} & P_{11} & \cdots & \cdots \\ \vdots & \vdots & & \\ \vdots & \vdots & & \end{pmatrix}$$

A square matrix P is called a stochastic matrix, if

- All matrix entries are nonnegative, and

- Each row sums to 1.

III. PAGERANK ALGORITHM

PageRank is a Google algorithm that measures the importance of web pages based on the quality and quantity of links pointing to them. It considers incoming links as votes, with pages receiving more high-quality links deemed more significant in search results.

In order to approximate the importance of a website, PageRank counts the quantity and caliber of links pointing to a certain page. The fundamental premise is that websites with greater authority will probably have more links coming from other websites.

A. Web Graph and Basic PageRank

Web graphs are useful to compute properties that need a global view. PageRank is one such property, but there are usually hundreds of similar properties. Web graphs from crawlers are at page level, called page graphs. By aggregating the vertices on some property, e.g., host, language, or domain, a hierarchy of web graphs can be created. A web graph $G=\{V,E\}$ s a directed, weighted graph whose vertices V correspond to web nodes and whose weighted edges E aggregate the hyperlinks of web pages in these nodes. G has an edge (u, v) if there is at least one hyperlink from a web page of node u to a web page of node v. Each vertex and edge can have one or more weights. For the purpose of this paper, the weight w(u, v) of an edge (u, v) is equal to the number of hyperlinks from pages of u to pages of v.

Besides its topological features, a web graph has properties. These properties can be numerical (scalars or distributions) or categorical (labels or lists). Some vertex properties that we will focus on are the PageRank of a vertex as a "quality" score, the list of websites pointing to the vertex (called its inlinks), and the list of websites the vertex is pointing to (called its outlinks).

The PageRank algorithm is built upon two fundamental ideas:
1. Endorsement through Links:
If webpage A links to web page B and webpage A has a high ranking, it is expected that this endorsement should contribute to a higher ranking for web page B.
2. Inverse Relationship with Out-Degree:
If webpage A links to multiple webpages, the influence or "vote" that A gives to each of these pages is inversely proportional to the total number of links on A. In other words, the more pages A links to, the less each individual link contributes to the ranking of the linked pages.

These principles capture the notion that a link from a highly-ranked page is more valuable than a link from a lower-ranked page, and that a page with fewer outgoing links (i.e., a lower out-degree) is more influential in boosting the rank of the pages it links to. The iterative nature of the PageRank algorithm calculates these relationships across the entire web graph, assigning each page a numerical score representing its relative importance or authority. The algorithm converges to a steady state, providing a ranking that reflects the overall importance of each page in the network.

PageRank algorithm uses only the link structure of the Web to determine the importance of a page rather than going into the contents of a page. PageRank provides a more efficient way to compute the importance of a Web page by counting the number of pages that are linking to it (in-coming links or backlinks). If an in-coming link comes from a reputed page, then that in-coming link is given a higher weight than those in-coming links from non-reputed pages. The PageRank PR of a page p can be computed by taking into account the set of pages pointing to p as per the formula.

$$PR(p)=\sum_{q\in pa_p}\frac{PR(q)}{O_q}\ (1)$$

Where:

- $PR(p)$: PageRank of a page p
- $PR(q)$: PageRank of a page q that points to page p
- $O(q)$: Number of outgoing links on page q

Let us take an example of a simple Web graph with three nodes 1, 2, and 3, as shown in Fig. 1.

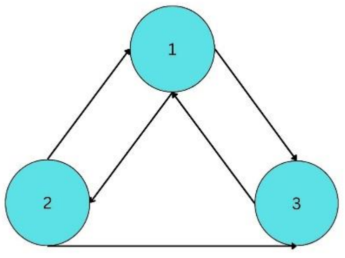


Fig. 1: Directed Graph

The PageRank for pages 1, 2 and 3 can be calculated by using (1). To start, we assume the initial PageRank rank $\frac{1}{n}$ and do the calculation.
Then we give all of its PageRank to all the nodes that it points to, and then we do this over and over again, performing these Basic PageRank Updating Rules k times. And then, the new value of PageRank for every node is

going to be simply the sum of all the PageRanks that are received from all the nodes that point to it.
.

PageRank ($k=1$)			
	A	B	C
Old	1/3	1/3	1/3
New	1/2	1/6	1/3

Crucially, we do the exact same thing again to get the second step of PageRank; k equals 2 in order to obtain the PageRank at Step 2 as shown in the figure below.

PageRank ($k=2$)			
	A	B	C
Old	1/2	1/6	1/3
New	5/12	1/4	1/3

From the figure above, after two steps, we find that node A has the highest PageRank (0.42), followed by node C, and then node B. This point suggests that A is the most important node in this network.

What happens if we continue for another step?

If we continue with more steps, we might notice that the values change a little bit, but they still have the same order, and A is still the highest PageRank node

B. *Scaled PageRank*

- The PageRank of a node at step k is the probability that a **random walker** lands on the node after taking k steps.
- **Random walk at k steps:** Start on a random node, because the probability of each node is the same. Then choose an outgoing edge at random and follow it to the next node. Repeat k times.
- For example, the graph in Fig. 2 will be used to show a random walk of 5 steps that ($k=5$) looks like this:

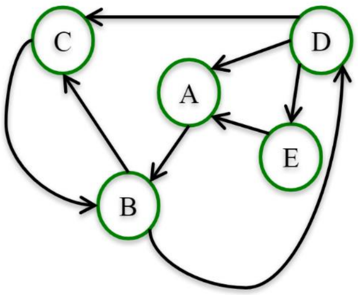


Fig. 2: A directed graph showing random walk.

- Now, begin the random walk.

- Step 1: Choose a random node. Then, choose a random outgoing edge and after that, follow the edge to the next node.
For example, node D is chosen to begin the random walk and follow the outgoing edge to node A.

- Step 2: Choose a random outgoing edge and follow it. Now we continue from node A; in this case, there are no options. The only one you can choose is the one going to node B.

- Step 3: Choose a random outgoing edge and follow it. And then you choose again a random edge, and there are two options: either go to C or go to D. In this situation, we go to D.

- Step 4: Choose a random outgoing edge and follow it. Then, out of C, there's only one edge; you have to go back to B, so you go back to B.

- Step 5: Choose a random outgoing edge and follow it. And then, when we're back at B, we choose again randomly between going to C and D. In this case, you went to D, so that is a random walk with 5 steps.

And so in thinking about this interpretation of PageRank that says that the value of PageRank of each node is the probability that would land on that node after k steps. Well, we computed the PageRank values for this network. If you repeat this for a lot of steps, say k equals infinity. These are the values that you can eventually approach, these are the values that you converges to.

- Here are the result:

PageRank					
	A	B	C	D	E
$k=\infty$	0.12	0.38	0.25	0.19	0.06

So here, B had the highest value of PageRank of 0.38. This value is the probability that are random walk after taking many, many, many steps would land on node B.

- Imagine a random walk on the graph on Fig. 3

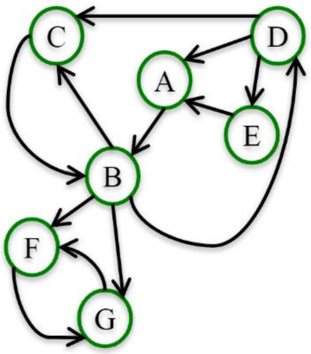


Fig. 3: A directed graph after added two nodes, F and G.

Start the random walk, beginning with node D. Whenever the random walk lands on F or G, which will happen eventually if we walk long enough on this network, then they're going to stock on F and G because there are no edges to go to. So, there's no way to get back from G and F to any of the other nodes.

And like all the other nodes, the probability that you land on one of them after taking a very, very long random walk is going to be 0. And the probability of landing on either F or G after a very long random walk is going to be about half for each.

- To fix this, we use the "**damping parameter**" α .

Random walk at k steps with damping parameter α: Start on a random node. Then:

- With probability α: choose an outgoing edge at random and follow it to the next node

- With probability $1-\alpha$: choose a node at random and go to it.

Repeat k times.

So the random walk is no longer "stuck" on nodes F and G.

- The Scaled PageRank of k steps and damping factor α of a node is the probability that a random walk with damping factor α lands on a node after k steps. For most networks, scaled k, PageRank converges to a unique value, which depends on α

- In practice and calculation, the damping factor α is set between 0.8 and 0.9, usually at 0.85.

- Here are the PageRanks of all the nodes from the graph in Fig. 3

Scaled PageRank ($\alpha = 0.8$, k large)						
A	B	C	D	E	F	G
0.08	0.17	0.1	0.08	0.05	0.27	0.25

F and G still have higher PageRank, but not all the PageRank

- The damping factor work better in very large networks like Web or larger social networks.

- To calculate exactly the Scaled PageRank value of graph, we can use the function $pagerank(G, alpha = 0.8)$ from the NetworkX library in Python.

IV. Using the Markov chain in Google PageRank algorithm

The Markov chain is not addressed in Brin et al.'s original PageRank method. However, the other researchers looked at how the Markov chain and the PageRank algorithm relate to one another. The connection between the Markov chain and the PageRank algorithm is explained in this section. Envision an arbitrary web surfer traversing the internet, selecting an outbound link at random from one page to the next. This occasionally results in dead ends, or pages that loop around a collection of related pages but have no outbound connections. Thus, the surfer selects a random website from the Web a specific percentage of the time. It is referred to as the Markov chain or Markov process, this hypothetical random walk. PageRank is the limiting probability that a randomly selected, endlessly committed surfer will visit a given page.

Problem: Implementing PageRank on famous social network.

Input:

1. Social Network Graph: The structure of the social network represented as a graph. This can be in the form of an adjacency matrix, adjacency list, or any other graph representation.

2. Damping Factor (Optional): The damping factor (d) represents the probability that a user will continue to another page rather than following a link. It is typically set to 0.85.

Output:

1. PageRank Scores: The output is a set of PageRank scores assigned to each node in the social network. These scores represent the importance or influence of each node within the network.

2. **Ranked List of Nodes:** A ranked list of nodes based on their PageRank scores, indicating the relative importance of each node in the social network.

Here are our steps to solve this problem:

1. Create an initial vector
2. Create a transition matrix.
3. Create a Google matrix
4. Calculate the PageRank.

We decided to apply the Google PageRank Algorithm to the graph on Fig. 3

A. *Initial Vector and Transition Matrix*

A minimum of one author is required for all conference articles. Author names should be listed starting from left to right and then moving down to the next line. This is the author sequence that will be used in future citations and by indexing services. Names should not be listed in columns nor group by affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization).

1) *Create Initial Vector:*

To create the initial vector for a graph, you assign probabilities to each node in a way that reflects an equal likelihood of starting the random walk from any node. The initial vector is a probability distribution where each node is assigned a probability of 1/n, which is the total number of pages on the Web. This uniform distribution ensures an unbiased starting point for the PageRank calculation, where the random walker has an equal chance of beginning the walk from any node in the graph.
The graph in Fig. 3 has the initial vector:

$$V = \frac{1}{n}\begin{pmatrix}1\\1\\1\\1\\1\\1\\1\end{pmatrix} = \begin{pmatrix}1/7\\1/7\\1/7\\1/7\\1/7\\1/7\\1/7\end{pmatrix}$$

2) *Transition Matrix*

The **transition matrix** M in Fig. 3 can be produced by applying the Markov chain theorem.

$$M = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/4 & 1/4 & 0 & 1/4 & 1/4 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 1/3 & 0 & 1/3 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

There are non-zero elements in row q and column p of the *transition matrix* M that correspond to forward and back links, respectively, to page q and page p, respectively. The row total equals one if page q contains forward links.

A page without any forward links is indicated by the transition matrix's indication that the sum of any row is 0. We refer to this kind of page as a *"hanging node"* or *"dangling node."* When presenting the Web graph with a Markov model, it is not possible to have any dangling nodes.

B. *Google Matrix*

According to Langville et al. [9], Brin and Page added a perturbation matrix $E = VV^t / n$ to make this stochastic irreducible matrix as Google matrix as shown in (2)

$$G = \alpha * M + (1-\alpha) * E \qquad (2)$$

Where:

- *V*: initial vector
- E: perturbation matrix
- α : damping factor.

This matrix computation can be normalized to a stationary vector by calculating the powers of the transition matrix. At one stage of the calculation, the values of the matrix get stationary.

Here is the Google Matrix base on transition matrix M:

$$G = \begin{pmatrix} 0.021 & 0.871 & 0.021 & 0.021 & 0.021 & 0.021 & 0.021 \\ 0.021 & 0.021 & 0.234 & 0.234 & 0.021 & 0.234 & 0.234 \\ 0.021 & 0.871 & 0.021 & 0.021 & 0.021 & 0.021 & 0.021 \\ 0.305 & 0.021 & 0.305 & 0.021 & 0.305 & 0.021 & 0.021 \\ 0.871 & 0.021 & 0.021 & 0.021 & 0.021 & 0.021 & 0.021 \\ 0.021 & 0.021 & 0.021 & 0.021 & 0.021 & 0.021 & 0.871 \\ 0.021 & 0.021 & 0.021 & 0.021 & 0.021 & 0.871 & 0.021 \end{pmatrix}$$

C. *Calculate the Pagerank Values*

The final PageRank value of the Web graph depend on number of iteration of the random walk using the formula:

$$PR_{k+1} = G * PR_k \qquad (3)$$

Where:

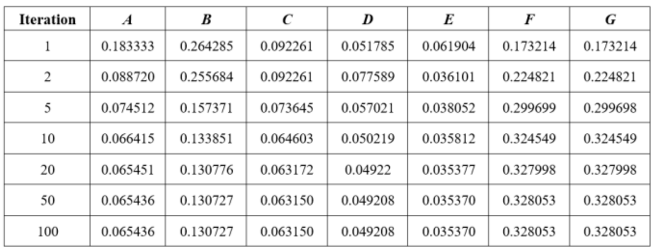
PR_k: The PageRank vector at iteration *k*

G: The Google Matrix

k: number of iteration

Here are the result:

TABLE I. PAGERANK VALUES OF FIG. 3



Iteration	*A*	*B*	*C*	*D*	*E*	*F*	*G*
1	0.183333	0.264285	0.092261	0.051785	0.061904	0.173214	0.173214
2	0.088720	0.255684	0.092261	0.077589	0.036101	0.224821	0.224821
5	0.074512	0.157371	0.073645	0.057021	0.038052	0.299699	0.299698
10	0.066415	0.133851	0.064603	0.050219	0.035812	0.324549	0.324549
20	0.065451	0.130776	0.063172	0.04922	0.035377	0.327998	0.327998
50	0.065436	0.130727	0.063150	0.049208	0.035370	0.328053	0.328053
100	0.065436	0.130727	0.063150	0.049208	0.035370	0.328053	0.328053

In iterative methods. Increasing the number of iterations usually results in a more precise estimate.

Using Python on Google Colab to calculate with 100 iterations to obtain the result vector as follows:

$$S = [0.0654; 0.1307; 0.0632; 0.0492; 0.0354; 0.3281; 0.3281]$$

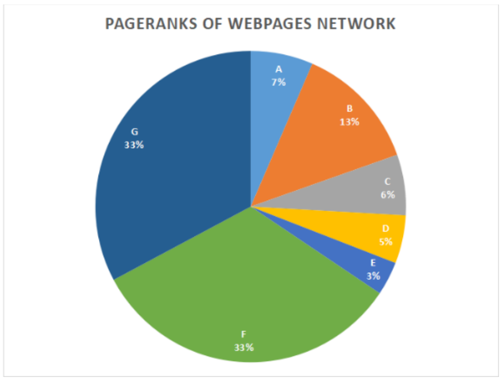


Fig. 4: PageRanks of Webpage Network

Compared to the result of the function pagerank(M,0.85) of the NetworkX library, we have the same result.

Based on the above results, we can rank each website. This is also the way Google ranks the relevance of a website.

V. CONCLUSION

This paper starts with the introduction of the Markov chain and PageRank algorithm. Then the mathematics behind the PageRank algorithm is explained theoretically. This paper also provides anonymity about how the PageRank algorithm uses the Markov chain and transition matrix to calculate the relevancy set. This paper highlights the different adjustments made to make the Web graph into a Markov model. In that, the dangling node problem and the methods to handle the dangling nodes were also discussed and the mathematical solutions were given. A Markov model is created for a sample Web graph and the PageRank calculation is shown for the Markov model. We implemented the PageRank algorithm just to support our mathematical model and showed the results.

We have only calculated with a 7x7 matrix, but in fact, when Google calculates PageRank for the entire World-Wide-Web, it can go up to a 14 billion x 14 billion matrix. Therefore, we will look for larger data sources to test and develop the program to be able to calculate with larger matrices and develop models for application in related fields and will be improved in the future.

REFERENCES

[1] Talay, D. (2003, September 1). Book Review: Finite Markov chains and algorithmic applications. *Mathematics of Computation*, *72*(243), 1573–1576.

[2] Papadimitriou, P., Dasdan, A., & Garcia-Molina, H. (2010, February 25). Web graph similarity for anomaly detection - Journal of Internet Services and Applications. SpringerLink.

[3] Xue GR, Yang Q, Zeng HJ, Yu Y, Chen Z (2005) Exploiting the hierarchical structure for link analysis. In: SIGIR, New York, NY, USA, pp 186–193.

[4] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Libraries SIDL-WP-1999-0120, 1999.

[5] "Facts about Google and Competition". Archived from the original on 4 November 2011. Retrieved 12 July 2014.

[6] Wikipedia contributors. (2023a, December 18). *Markov chain*. Wikipedia. https://en.wikipedia.org/wiki/Markov_chain

[7] Kumar, Ravi & Singh, Ashutosh & Goh, Kwang Leng. (2011). Application of Markov Chain in the PageRank Algorithm.

[8] Hamra G, MacLehose R, Richardson D. Markov chain Monte Carlo: an introduction for epidemiologists. Int J Epidemiol. 2013 Apr;42(2):627-34. doi: 10.1093/ije/dyt043. PMID: 23569196; PMCID: PMC3619958.

[9] Langville, A. N., & Meyer, C. D. (2004). Deeper inside pagerank. Internet Mathematics, 1(3).

[10] A. Borodin, G.O. Roberts, J.S. Rosenthal and P. TsaParas, “Link Analysis Ranking: Algorithms, Theory and Experiments”, In Proc. Of the ACM Transactions on Internet Technology, Vol. 5, No 1, pp 231-297, 2005.

[11] Liu, B., & Liu, B. (2011). Social network analysis. Web data mining: Exploring hyperlinks, contents, and usage data, 269-309.

[12] Chung, F. (2014). A Brief Survey of PageRank Algorithms. IEEE Trans. Netw. Sci. Eng., 1(1), 38-42.

[13] Rogers, I. (2002). The Google Pagerank algorithm and how it works.