PageRank Unveiled

Demystifying the Algorithm Behind the Trillion-Dollar Search Industry

Minor in AI, IIT Ropar

19th April, 2025

A Beginner's Guide to Understanding the Magic of Search Engines

1 The World Wide Web & Importance of Ranking

Imagine you're searching for the perfect chocolate chip cookie recipe online. You type your query into Google, and *bam!* - millions of results appear. But you don't sift through all of those pages, do you? You're likely to click on one of the first few links, hoping they hold the secret to that perfect, gooey, delicious cookie.

Ever wondered how Google decides which pages to show you first? It's not random chance. It's all thanks to a clever algorithm that revolutionized the internet and launched a trillion-dollar industry. This algorithm is called **PageRank**, and this book is going to explain how it works.

1.1 Why is Ranking Important?

Think about it: the internet is a vast ocean of information. Without a way to organize and prioritize it, finding what you need would be like searching for a single grain of sand on a beach. PageRank is the compass that guides us through this ocean, pointing us to the most relevant and trustworthy sources. It allows anyone to easily search for "Cooking Recipes" and go through it without searching for individual websites. Without the search engines, this task would be very tedious and unmanageable.

2 The Web as a Graph

To understand PageRank, we need to think of the internet in a new way: as a **graph**.

- **Pages:** Each webpage is a node in the graph. Let's call three webpages as 'A', 'B' and 'C'.
- **Links:** The hyperlinks connecting pages are the edges in the graph. If page A links to page B, we draw an arrow from A to B, indicating that a user can navigate from A to B by clicking on the link.

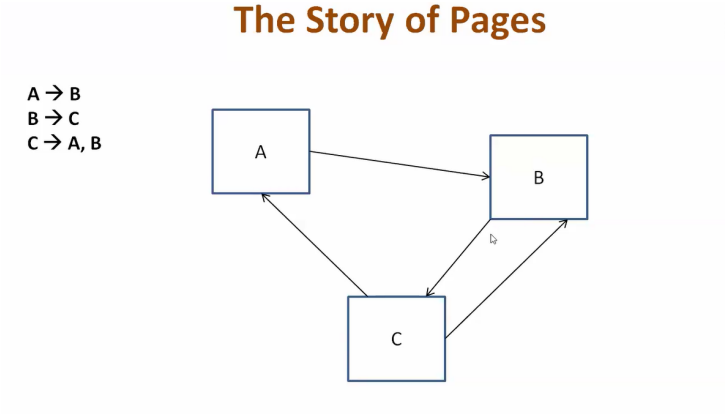

Figure 1: A simple graph with three nodes (A, B, C) and arrows connecting them

Imagine the three webpages A, B and C are linked together in the following way:

- A links to B.
- B links to C.
- C links to A and B.

Now, let's define two important concepts:

- **In-degree:** The number of links pointing *to* a page. In our example:
  - A has an in-degree of 1 (from C).
  - B has an in-degree of 2 (from A and C).
  - C has an in-degree of 1 (from B).
- **Out-degree:** The number of links pointing *out* of a page. In our example:
  - A has an out-degree of 1 (to B).
  - B has an out-degree of 1 (to C).
  - C has an out-degree of 2 (to A and B).

These concepts are crucial to understanding how PageRank assigns importance to different web pages. Intuitively, a page with a high in-degree (many links pointing to it) is considered more important, because many other pages deem it valuable enough to link to.

3 The Random Surfer Model

Now, imagine a user browsing the internet, clicking on links randomly. This is known as the **Random Surfer Model**. This model makes two key assumptions about user behavior:

1. **Following Links:** The surfer clicks on a link on the current page and goes to the new page.

2. **Teleportation:** The surfer gets bored and jumps to a completely random page on the internet. Maybe they were looking for a chocolate chip cookie recipe, but now they want to know about the history of baking soda!

These assumptions are quantified by the concept of **Damping Factor**.

4 Damping Factor

The Damping Factor (represented by 'd') is the probability that the random surfer will *continue* clicking links on the current page. This value is, in practice, set between 0 and 1. The opposite of Damping Factor is called "Teleportation". In layman's terms, Damping Factor is used to signify the probability of a user being "interested" enough to follow a link from the current webpage. Teleportation is the process by which the user gets bored and leaves the current webpage to visit a completely unrelated webpage. This process is necessary to ensure that all webpages have a fair probability of being visited; even those that have very low ratings.

Google's original PageRank algorithm uses a damping factor of 0.85 (85%). This means that there's an 85% chance the surfer will continue clicking links, and a 15% chance they'll jump to a random page.

Note

The two co-creators of PageRank, **Larry Page** and **Sergey Brin**, arrived at this number through experimentation and analysis of user behavior. They tried different values and found that 0.85 seemed to give the best results in terms of ranking quality.

5 Calculating PageRank - The Iterative Process

Now, let's get to the core of PageRank: the algorithm itself. The page rank of a webpage 'A' can be calculated using the page rank of other webpages that link to it. The more webpages that point to webpage A, the higher the rank of webpage A is likely to be. Also, if the webpages that link to webpage A are themselves highly ranked, then the page rank of webpage A is also likely to increase. To simplify calculations, we'll assume that the total rank of all pages on the web is 1. We can then divide this score amongst all the webpages according to their contribution to the web.

The PageRank algorithm works iteratively. It starts by assigning an initial rank to each page. Then, it repeatedly updates these ranks based on the links between pages until the ranks converge (stop changing significantly).

Here's how the iterative process works:

1. **Initialization:** Initially, all webpages in the graph are given the same rank. The rank of each webpage would then be (1 / N), where 'N' is the number of webpages on the internet. So, in our case, let's say that the three webpages 'A', 'B' and 'C' are initialized with the same rank of 0.33 each (33%).

2. **The Constant Part:** Each webpage is given a constant rank score, which is calculated using the formula: (1 - d) / N, where 'd' is the damping factor and 'N'

is the number of webpages on the internet. This score is intended to ensure that all webpages have a baseline rank score; even those that have zero in-degree. Thus, the constant rank score for each webpage would be equal to: (1 - 0.85) / 3 = 0.05. So, each webpage would be given an inherent rank score of 0.05.

3. **Iteration:** For each iteration, the page rank is updated by the following formula (for each node 'A'):

$$PR(A) = \frac{1-d}{N} + d \times \left(\frac{PR(T_1)}{C(T_1)} + ... + \frac{PR(T_n)}{C(T_n)} \right) \quad (1)$$

Where:

- $PR(A)$ is the PageRank of page A.
- d is the damping factor (0.85).
- N is the total number of pages.
- T_1 to T_n are the pages that link to page A.
- $PR(T_i)$ is the PageRank of page T_i.
- $C(T_i)$ is the out-degree of page T_i.

Translation: The PageRank of page A is the constant rank score + the sum of ranks of all other pages ($T_1...T_n$) that point to the current page 'A', after being reduced by their out-degrees. It is then scaled by the Damping Factor. Let's apply this to our three webpages:

- **A:** Has the initial page rank of 0.33. It has an in-degree of 1: (C). The constant rank score for the page is: 0.05. Therefore, the new page rank of A will be:
  - $PR(A) = 0.05 + 0.85 \times (0.33/2) = 0.1915$.
- **B:** Has the initial page rank of 0.33. It has an in-degree of 2: (A, C). The constant rank score for the page is: 0.05. Therefore, the new page rank of B will be:
  - $PR(B) = 0.05 + 0.85 \times (0.33/1 + 0.33/2) = 0.4745$.
- **C:** Has the initial page rank of 0.33. It has an in-degree of 1: (B). The constant rank score for the page is: 0.05. Therefore, the new page rank of C will be:
  - $PR(C) = 0.05 + 0.85 \times (0.33/1) = 0.3305$.

We will iterate the process one more time:

- **A:** Has the initial page rank of 0.1915. It has an in-degree of 1: (C). The constant rank score for the page is: 0.05. Therefore, the new page rank of A will be:
  - $PR(A) = 0.05 + 0.85 \times (0.3305/2) = 0.191525$.

- **B:** Has the initial page rank of 0.4745. It has an in-degree of 2: (A, C). The constant rank score for the page is: 0.05. Therefore, the new page rank of B will be:
  - $PR(B) = 0.05 + 0.85 \times (0.191525/1 + 0.3305/2) = 0.3543$.
- **C:** Has the initial page rank of 0.3305. It has an in-degree of 1: (B). The constant rank score for the page is: 0.05. Therefore, the new page rank of C will be:
  - $PR(C) = 0.05 + 0.85 \times (0.4745/1) = 0.453325$.

4. **Convergence:** The algorithm continues iterating until the PageRank values stabilize, meaning they don't change much from one iteration to the next. This indicates that the algorithm has reached a stable ranking of the pages.

6 Python Code Example

Here's a simplified Python code example to illustrate the PageRank algorithm:

```
import numpy as np

def pagerank(link_matrix, damping_factor=0.85, max_iterations=100):
    """
    Calculates PageRank for a set of webpages.

    Args:
        link_matrix: A matrix representing the links between pages.
                     link_matrix[i][j] = 1 if page j links to page i, 0 otherwise.
        damping_factor: The probability that the surfer will follow a link.
        max_iterations: The maximum number of iterations to perform.

    Returns:
        A numpy array containing the PageRank scores for each page.
    """

    num_pages = link_matrix.shape[0]
    rank_vector = np.ones(num_pages) / num_pages  # Initial rank vector

    for _ in range(max_iterations):
        new_rank_vector = np.zeros(num_pages)
        for i in range(num_pages):
            new_rank_vector[i] = (1 - damping_factor) / num_pages
            for j in range(num_pages):
                if link_matrix[i][j] == 1:
```

```
                    out_degree = np.sum(link_matrix[:, j])
                    new_rank_vector[i] += damping_factor * (
                        rank_vector[j] / out_degree)

        # Check for convergence (if the rank vector doesn't
            change much)
        if np.allclose(new_rank_vector, rank_vector):
            break

        rank_vector = new_rank_vector

    return rank_vector

# Example usage:
link_matrix = np.array([
    [0, 0, 1],  # A is linked to by C
    [1, 0, 1],  # B is linked to by A and C
    [0, 1, 0]   # C is linked to by B
])

page_ranks = pagerank(link_matrix)
print("PageRank scores:", page_ranks)
```

This code implements the iterative PageRank algorithm described earlier. The `link_matrix` represents the connections between pages, and the `pagerank` function calculates the PageRank scores based on this matrix and the damping factor. The result is shown for a maximum of 100 iterations, or whenever convergence occurs.

7 Limitations and Beyond

The PageRank algorithm, while groundbreaking, has some limitations. It primarily focuses on the *quantity* of links, not the *quality*. A page could artificially inflate its rank by acquiring many low-quality links. Today's search algorithms are far more sophisticated, taking into account factors such as:

- Content relevance
- Website quality
- User experience
- Mobile-friendliness
- And many more!

These complex algorithms are updated frequently, constantly evolving to provide the best possible search results. Also, it is nearly impossible for the PageRank algorithm to be manually applied on the World Wide Web.

Note

PageRank was a crucial step in the evolution of search engines, and its core principles continue to influence how search results are ranked today. By understanding PageRank, you gain valuable insights into the inner workings of the internet and the trillion-dollar search industry.