PageRank

Ben Burns, Dan Magazu, Lucas Chagas,
Thomas Webster, Trung Do

Fall 2021

Table of Contents

Motivation

- Problem: the internet has a *lot* of web pages
- A lot of the information out there either isn't relevant to us, or is inaccurate
- Motivation: we want a program that, when provided a phrase, returns webpages with information relevant to the input
- Intuitive solution: return back websites that either contain that phrase, or contain similar phrases
- This helps us find more *relevant* pages, but we can't know if we're getting the best information (much less *accurate* information) without manually going through each result
- We desire a stronger solution

PageRank

- Invented by Sergey Brin and Larry Page (1998)[1]
  - Publication marks them becoming co-founders of Google
- Idea: we want some way to numerically score each webpage based on how "important" it is
- Algorithm numerically scores each page p based on
  - How many other pages link to p (or "cite" it)
  - The "importance" of each of p's citations
- We then numerically order pages to rank them
- PageRank: the procedure for scoring each website
- Google: the database that indexes the PageRank of each website for search

[1] Many use the year of the original manuscript, 1996

Underlying Assumptions

- Running our basic search engine gives us a collection of pages with information relevant to our query
- Assumption: More "important" and useful websites will be the ones with proportionally more inbound links
- Pages with very reliable, primary information are likely to be cited by lots of website authors, and therefore will have lots of "flow" into them
- Assumption: websites with no outgoing links are treated as linking to every other website
- Else, these pages would just take in flow without ever redistrubting, and eventually all users get stuck on these pages

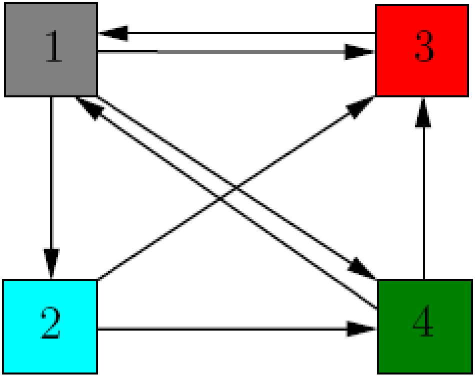
Finding Less Important Information

- A worry you may have is that we'll only find pages with lots of inbound links
- You can still find niche information by making your query more specific so that it won't match more general pages
- Searching "PageRank" will likely get you Wikipedia, but "Anatomy of PageRank architecture" gives you the original research literature.
- Your returned urls are still proportionally important results, your query just filtered out the numerically more "important", yet less relevant pages.

Formalizing the PageRank problem

- We're going to construct a directed graph $G = (V, E)$
- For each website we consider, we construct a node $v_i \in V$
- For two distinct nodes $v_i,\ v_j \in V$, the *directed* edge $v_i v_j \in E$ iff there is a link on website i that goes to website j.
- If v_i and v_j are not distinct (a website is linking to itself), we ignore the link and do not construct a loop edge.
  - G is not a psuedograph
- Multiple hyperlinks on page i to page j are all represented by the single, directed edge
  - G is not a multigraph.

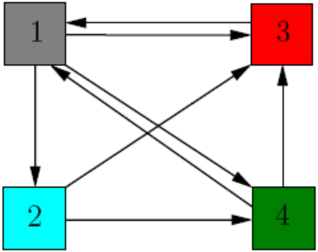
Visual Representation



- We have a set of 4 websites
- Each edge represents a hyperlink from the origin node to the destination node

Applying PageRank Values

- At first, we assign each vertex $v \in V$ with a weight of $\frac{1}{|V|}$.
- All vertices have equal weight, and our weights sum up to 1. The weight of a particular vertex v_i is denoted $PR(v_i)$.
- Consider the set V_i of vertices that v_i has an edge to:
  - $V_i = \{v_j | v_i v_j \in E\}$
- A user on page 1 can choose to click a link to traverse to either 2, 3, or 4. In other words, $V_1 = \{2, 3, 4\}$.



Traversing from Page to Page

- Assumption: A user on page i user is equally likely to choose to visit each vertex in V_i (our set of vertices that v_i cites)
- In our example from before, the probabilty that our user on page 1 visits page 2 is $P(v_2) = \frac{1}{|V_1|} = \frac{1}{3}$
- So a third of page 1's visitors will "flow" to page 2, a third to page 3, and a third to page 4
- From v_i, $P(v_j) = \frac{1}{|V_i|}$ if $v_j \in V_i$, and 0 else.

Iteration Model

- All at once, all users will click one of the links on their current page
- At "click 0", the PageRank value of all vertices $PR(v) = \dfrac{1}{|V|}$
- On click 1, each page will equally split its PageRank value among the pages it cites
- Page 1 starts with $PR(v_1) = \dfrac{1}{4}$ at click 0, and contributes $\dfrac{1}{12}$ to each of 2, 3, and 4 on click 1
- Conversely, page 1 will receive nothing from page 2, all of $PR(v_3)$, and half of $PR(v_4)$. So after click 1, $PR(v_1) = 0 \cdot PR(v_2) + PR(v_3) + \dfrac{1}{2}PR(v_4) = \dfrac{1}{4} + \dfrac{1}{8} = \dfrac{3}{8}$

Example iterations

PR	0	1	2	3	4	5	6	7
v_1	$\frac{1}{4}$	$\frac{3}{8}$	$\frac{7}{16}$	0.3542	0.3958	0.3906	0.3819	0.3898
v_2	$\frac{1}{4}$	$\frac{1}{12}$	$\frac{1}{8}$	0.1458	0.1181	0.1319	0.1302	0.1273
v_3	$\frac{1}{4}$	$\frac{1}{3}$	$\frac{13}{48}$	0.2917	0.2951	0.2865	0.2917	0.2905
v_4	$\frac{1}{4}$	$\frac{5}{24}$	$\frac{1}{6}$	0.2083	0.1910	0.1910	0.1962	0.1924

- Looks okay, but the values don't quite converge

Introducing a Damping factor

- At some point, our user who is randomly clicking links will eventually stop clicking links
- At each iteration we consider a constant probability $d :=$ the probability that our user stops traversing the graph
  - Various studies have settled on $d \approx 0.85$
- For an arbitrary graph:

$$PR(v_i) = \frac{1-d}{|V|} + d(c_1 PR(v_1) + c_2 PR(v_2) + c_3 PR(v_3))$$

- $c_n = \dfrac{1}{|V_n|}$ if $i \in V_n$, and 0 else.
  - If n cites i, then n contributes its equally distributed value to i. If n doesn't cite i, n contributes 0 to i.
- If $|V_j| = 0$, we use $|V|$ (random jump from sinks)

Iterations w/ Damping

PR	0	1	2	3	4	5	6
v_1	$\frac{1}{4}$	0.3562	0.4014	0.3502	0.3720	0.3697	0.3664
v_2	$\frac{1}{4}$	0.1083	0.1384	0.1512	0.1367	0.1429	0.1422
v_3	$\frac{1}{4}$	0.3208	0.2757	0.2885	0.2903	0.2864	0.2884
v_4	$\frac{1}{4}$	0.2146	0.1845	0.2101	0.2010	0.2010	0.2030

Applications

- PageRank is perhaps the most famous search ranking algorithm. Googles high-quality search engine results are directly correlated to PageRank
- There are a remarkable wide variety of applications of the PageRank algorithm that apply to non-search engine contexts.
- Its simplicity and elegance allow PageRank to be a more general and powerful tool.
- Let's look at the applications of PageRank and its connection to Twitter.

Twitter

- In 2010, Twitter was lagging behind and was lacking a user recommendation service.
- This was perceived both externally and internally as a critical gap in Twitter's product offerings, so quickly launching a high-quality product was a top priority.
- Twitter is unique because of the asymmetric nature of the following relationship—a user can receive messages from another without reciprocation.
- This differs substantially from other social networks such as Facebook or LinkedIn, where social ties can only be established with the consent of both participating members.

Introducing PageRank

- This works well with PageRank because we can determine outbound and inbound links.
- A user u is likely to follow those who are followed by users that are similar to u.
- These users are in turn similar to u if they follow the same (or similar) users.
- Therefore using Page Rank, Twitter is able to offer unique recommendations for users.
- By analyzing who the user follows and who those users follow, the PageRank algorithm will allow Twitter to make specific recommendations for each user.
- Essentially, the more outbound links that an account receives correlate to a higher PageRank score.