Data-analysis and Retrieval
PageRank

Hans Philippi

May 8, 2024

Outline

- Introduction
- Recap linear algebra
- Basics of Pagerank
- About the existence of a solution and the convergence of the algorithm

A short history

- 1995-2000: known techniques from IR applied to WWW
- Classical approach: focus on relevance of document to query
- Problem: many many answers
- Average user will look at 10-20 answers at most
- Growing insight: need to distinguish "important" sites

A short history

- 1998 John Kleinberg (IBM Almaden) tries to use the hyperlink structure of the web
- At the same time, Sergey Brin en Larry Page are developing the PageRank algorithm at Stanford University
- Question: can we use the link structure of the web to identify important sites

Principles of "importance" in PageRank

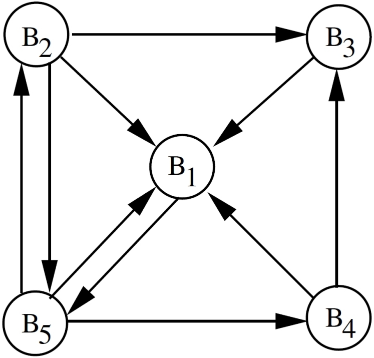
Using link structure to define importance of a web site:

- When many sites refer to you, you are important
- When important sites refer to you, you are important [1]
- When a site referring to you has many outgoing links, this decreases the weight of the reference

[1] This feels like a circular definition, but we can deal with it!

The web is a directed graph

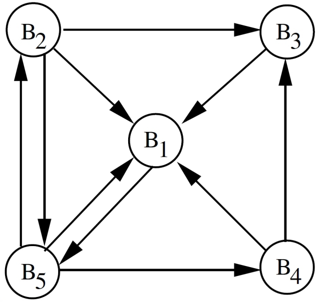
- The set of nodes corresponds to web sites: B_i
- The set of links corresponds to the hyperlinks



PageRank: model & math

- Each site B_i has a value P_i, the *pagerank*
- The web induces a set of equations for the pageranks P_i

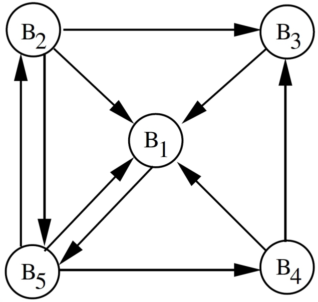
$$
\begin{aligned}
P_1 &= P_2/3 + P_3/1 + P_4/2 + P_5/3 \\
P_2 &= \dots \\
P_3 &= \dots \\
P_4 &= \dots \\
P_5 &= \dots
\end{aligned}
$$



PageRank: model & math

- Each site B_i has a value P_i, the *pagerank*
- The web induces a set of equations for the pageranks P_i

$$
\begin{aligned}
P_1 &= P_2/3 + P_3/1 + P_4/2 + P_5/3 \\
P_2 &= P_5/3 \\
P_3 &= P_2/3 + P_4/2 \\
P_4 &= P_5/3 \\
P_5 &= P_1/1 + P_2/3
\end{aligned}
$$



- We can reformulate the set of equations in terms of vectors and matrices
- $HP = P$, with

$$H = \begin{bmatrix} 0 & \frac{1}{3} & 1 & \frac{1}{2} & \frac{1}{3} \\ 0 & 0 & 0 & 0 & \frac{1}{3} \\ 0 & \frac{1}{3} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{3} \\ 1 & \frac{1}{3} & 0 & 0 & 0 \end{bmatrix} \qquad P = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \end{bmatrix}$$

Intermezzo

$$H = \begin{bmatrix} 0 & \frac{1}{3} & 1 & \frac{1}{2} & \frac{1}{3} \\ 0 & 0 & 0 & 0 & \frac{1}{3} \\ 0 & \frac{1}{3} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{3} \\ 1 & \frac{1}{3} & 0 & 0 & 0 \end{bmatrix}$$

- Calculate the sum of each column. What do you notice? Explain.
- Does the same hold for each row? Explain.

PageRank: model & math

- You can solve this set of equations using standard techniques from linear algebra
- But we have this huge dimension: $n =$ number of sites, about 10^{10}
- The complexity of the solving algorithm is $O(n^3)$
- The algorithm has no approximation behaviour: it is all or nothing

Alternative calculation: *fixpoint iteration*

- Start with a vector $P^{(0)} = (1/n, 1/n, ..., 1/n)^T$
- Calculate $P^{(k)} = HP^{(k-1)}$, for a certain k
- Hope (for this moment) that it converges towards a solution

Let us have a look at our toy example:

- $P^{(2)} = (0.3111, 0.0889, 0.0556, 0.0889, 0.4556)^T$
- $P^{(30)} = (0.3137, 0.1176, 0.0980, 0.1176, 0.3529)^T$
- It turns out to be the case that $P^{(30)}$ approaches the solution up to four decimals

PageRank: model & math

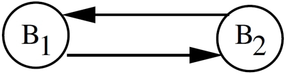
- Can we understand and guarantee the existence of a solution?
- Can we understand and guarantee the convergence of the fixpoint iteration toward the solution?
- Fortunately, we know some things from classical linear algebra
- The calculation is an *eigenvalue problem*:

$$HP = \lambda P$$

- There has been done a lot of research on eigenvalue problems in the previous century
- We will return to these questions soon

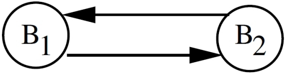
PageRank: some problems

- Can we guarantee the convergence of the fixpoint iteration?



PageRank: some problems

- Can we guarantee the convergence of the fixpoint iteration?

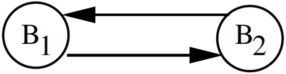


$$H = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

$$P^{(0)} = \begin{pmatrix} p_1 \\ p_2 \end{pmatrix};\ P^{(1)} = \begin{pmatrix} \dots \\ \dots \end{pmatrix};\ P^{(2)} = \begin{pmatrix} \dots \\ \dots \end{pmatrix};$$

PageRank: some problems

- Can we guarantee the convergence of the fixpoint iteration?

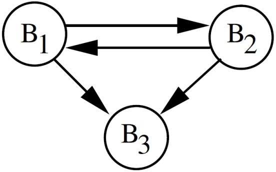


$$H = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

$$P^{(0)} = \begin{pmatrix} p_1 \\ p_2 \end{pmatrix};\ P^{(1)} = \begin{pmatrix} p_2 \\ p_1 \end{pmatrix};\ P^{(2)} = \begin{pmatrix} p_1 \\ p_2 \end{pmatrix};\ P^{(3)} = \begin{pmatrix} p_2 \\ p_1 \end{pmatrix}$$

PageRank: some problems

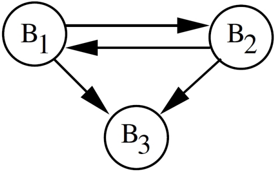
- Can we guarantee the existence of a useful solution?



$$H = \begin{pmatrix} 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}$$

Some problems

- Can we guarantee the existence of a useful solution?



$$H = \begin{pmatrix} 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}$$

The only solution is $P = (0, 0, 0)^T$

Toward convergence: teleportation

- B_3 has no outgoing edges; it is called a *dangling node*
- We will adapt our model with two goals in mind
- Goal 1: we need a mathematical trick to guarantee convergence to a useful solution
- Goal 2: the trick should make sense when modeling surfing behaviour
- The edges of the web graph correspond to clicking on links ...
- ... but sometimes, we just type an address or use a bookmark
- We model this phenomenon by *teleportation*

Toward convergence: teleportation

If our matrix H contains an empty column, we fill this column uniformly with values $1/n$

$$S = \begin{bmatrix} 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & \frac{1}{3} \\ 0 & 0 & \frac{1}{3} \\ 0 & 0 & \frac{1}{3} \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & 0 & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{3} \end{bmatrix}$$

In general:

$$S = H + \frac{1}{n} e a^T$$

with $e = (1, 1, 1, ..., 1)^T$ and $a_j = 1$
for each empty column j in H, otherwise 0

Toward convergence: teleportation

- We have used teleportation to solve the dangling node problem ...
- ... but it turns out that teleportation is the key to convergence in general!
- We will extend our model with a general notion of teleportation

Toward convergence: teleportation

- When someone is surfing, she will click on a link with a probability α, ...
- ... or type an new URL (teleport) with a probability $1 - \alpha$
- In our model, teleportation is a jump towards any known site, with uniform probability, represented by the teleportation matrix T

$$T = \frac{1}{n} e e^T = \frac{1}{n} \begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \dots & \dots & \dots & \dots \\ 1 & 1 & \dots & 1 \end{pmatrix}$$

[1] Please be aware of the dual role of the letter T

The Ultimate Equation

$$\boxed{G = \alpha S + (1-\alpha)T}$$

- We model clicking and jumping
- When $\alpha = 1$, we cannot guarantee convergence
- When $\alpha = 0$, we get results that completely ignore the structure of the web: all pages are equal
- In practice α is chosen close to 1 (0.85 is often suggested)
- However the closer to 1, the slower the convergence

Background: about convergence

The linear algebra behind PageRank was already known since more than a hundred years ago. It has been applied in the context of random walks, or, more specific, Markov chains.

- We have an array of stochastic variables $X_j, \ j = 0, 1, 2, \ldots$ representing a series of states
- Each B_i corresponds to a possible state
- The link matrix H corresponds to probabilities of state transitions $B_i \to B_j$

Example Markov chain

A game player can have three possible states: *playing*, *eating*, *sleeping*. The first column describes the transition probabilities for one step in time from status = playing. Keep playing: 0.92. From playing to eating: 0.05. From playing to sleeping: 0.03. Second column: from eating to playing: 0.7 etc.

$$H = \begin{pmatrix} 0.92 & 0.7 & 0.35 \\ 0.05 & 0.1 & 0.05 \\ 0.03 & 0.2 & 0.6 \end{pmatrix}$$

Note that each column sums up to 1.

Analogy between PageRank and Markov chains

- Starting with $P^{(0)} = \begin{pmatrix} 1/n \\ 1/n \\ ... \\ 1/n \end{pmatrix}$

  a repeated calculation $P^{(k)} = HP^{(k-1)}$ gives us a vector $P^{(k)}$ where each $P_i^{(k)}$ represents the propability of being in state B_i after a random walk of k steps, starting anywhere.
- Under certain conditions, $P^{(k)}$ converges to P^*, where each P_i^* approximates the probability of being in state B_i after a long random walk, starting anywhere.
- Analogy with PageRank: probability of status B_i corresponds to importance site B_i.

Background: about convergence

Now we will show how the theory behind random walks can be used to prove the convergence of the PageRank fixpoint algorithm.

Definition: a *probability vector* is a vector $u = \begin{pmatrix} u_1 \\ u_2 \\ \dots \\ u_n \end{pmatrix}$ such that each $u_i \geq 0$ and $\sum_{i=1}^{n} u_i = 1$

Definition: a *probability matrix* is a matrix where each column is a probability vector.

Background: about convergence

Definition: a probability matrix A is a *regular* when there exists a k such that A^k has only positive entries.

Check the following claims for the PageRank matrix G:

- G is a probability matrix
- G is a regular matrix (for $k =?$)

Background: about convergence

Theorem (Perron-Frobenius):
If A is regular probability matrix, then $\lambda = 1$ is an eigenvalue (the *principal eigenvalue*). For all other eigenvalues, $|\lambda| < 1$ holds.

So we have eigenvalues $\lambda 1, \lambda 2, \lambda 3, \ldots$ with

$$\lambda 1 = 1, |\lambda 2| < |\lambda 1|, |\lambda 3| < |\lambda 2|, \ldots$$

Corollary:
Because $\lambda = 1$ is an eigenvalue, we have a solution for $AP = P$

Theorem:
For regular probability matrices, $|\lambda 2|$ determines the speed of convergence.

Theorem:

For the PageRank matrix G:

$$|\lambda 2| \approx \alpha$$

Background: about convergence

Corollary: for the PageRank fixpount calculation with solution P, we have

$$||P - P^{(k)}|| \leq \ \alpha^k \ ||P - P^{(0)}||$$

- Note that for $\alpha = 0.85, \ \alpha^{50} \approx 0.0003$.
- This guarantees a precision of around three decimals for the calculated PageRank vector.

Conclusion: it works.

References

- Jan Brandts, *Over de wiskunde die Google groot maakte*
- Langville & Meyer, *Google's PageRank and Beyond*