L23: PageRank

Jeff M. Phillips

April 13, 2020

Final Report

At most 4 pages/student. Don't cram in too much!

- Succinct title (and names) ← Same for Posters
- Problem definition and motivation.
- Explain your Data.
- **key idea**
- What did you do (which techniques, an implementation, a comparison, an extension)
- What did you learn? Artifacts (charts, plots, examples, math) and Intuition (in words, did it work?)

Webpage Similarity (Search)

- **Inverted Index**

query "aple" — Edit dist → "apple" → page 1, p2, p3, ... (top 3) ... (top 10)

"boat" → page 7, p3,

"car" →

← sorted lists

page (title, url)

Define most relevant webpages

query "apple"

Jaccard
Set {apple}

Cosine
pie
apple
free

page
title — k-grams → min hash sig — d = Jac
text
apple apple apple — bag-of-words → word vector — d = Cos

help index (search for term, copy to ranked web pages)

Crawlers: program that walks around web: (1) read page
update feature vector

(2) follow random hyperlinks

inverted index ranking

use hyperlink info

<a href "www.pie.com"> pie

Spammers

build fleed pages: link to your page w/ hyperlink tag.

- <u>Indexes</u> : Alternative to search Engine

Yahoo! and LookSmart

Built an organized, curated collection of websites

Page Rank

$S(p_j, \text{term}) = f(\text{text}(p_j), \text{links to } p_j, q_*(j))$

- Pages are important if linked to by other important webpages.

delicate balance

- page is important if a "random surfer" were to find it.

random MCMC

Web is a big graph $G = (V, E)$

$V = \{\text{set of all pages}\}$

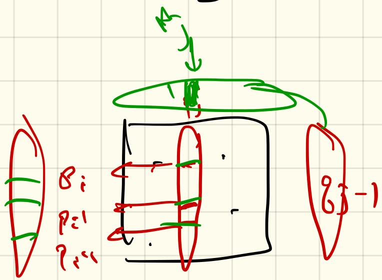
$E = \{E_{ij} = \text{link } p_i \to p_j\}$

Define MC $\to q_*$ ⇐ converged to vector distribution

$q_*(j)$ says how important page j is.

Compute q^* of Webgraph

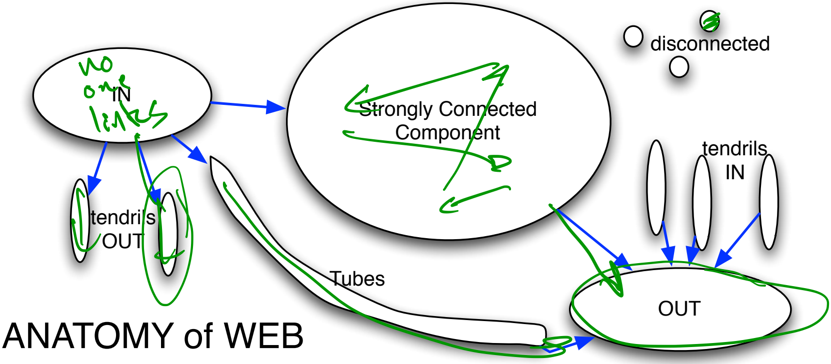
- Keep track of crawlers: how frequent return.
- Buy big computer: Compute eig(P)
  ↳ probtran(G)
- Precompute $P^* = P \cdot P \cdot P \cdot \ldots \cdot P$
  ↳ too big
- $q^* = q_0$ ← last night
  for $j = 1$ to 50
  $q_j = P q_{j-1}$



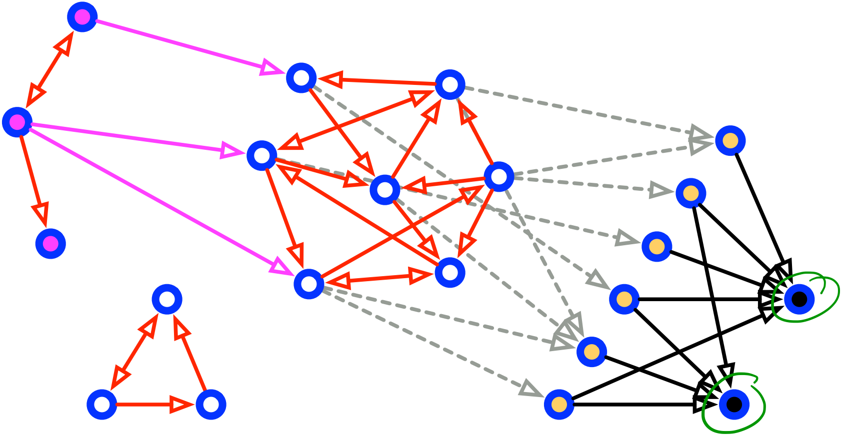
power method

Anatomy of Web

is this G ergodic



Anatomy of Web



Can we make G ergodic?

- Teleportation / taxation
  - → about once every 7 steps
    - → jump to random node.

P prob trans (G) $\qquad \beta = 0.15$

$$R = (1-\beta)P + \beta Q \qquad Q = \frac{1}{n}\begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$$

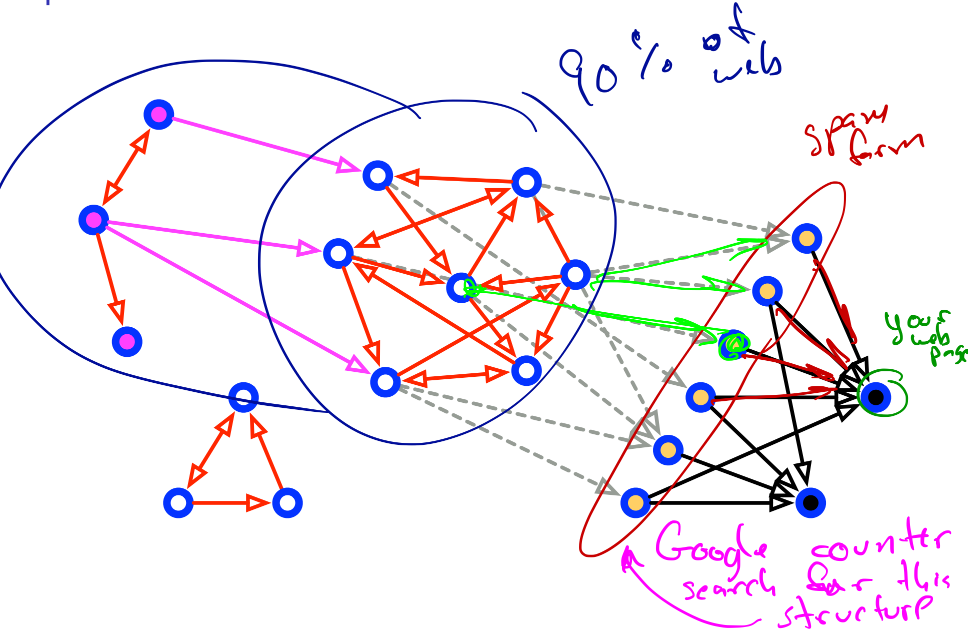
↳ dense

$$R q_i = ((1-\beta)P + \beta Q) q_{i-1}$$

$$(1-\beta) P q_{i-1} + \beta \mathbb{1}/n$$

$n \times 1$ vector

Spam Farms



90% of web
spam farm
your web page
Google search counter for this structure

Trust Rank (2015 ?)

Only teleport to trusted pages.

$r \leftarrow$ $g*$ pagerank

$t \leftarrow$ $g*$ trusted teleport

$$\frac{r(j) - t(j)}{r(j)}$$ if large → spam

↳ truthfulness of webpage

Word Count

Consider as input all of English Wikipedia stored in DFS. Goal is to count how many times each word is used.

Inverted Index

Consider as input all of English Wikipedia stored in DFS. Goal is to build an index, so each word has a list of pages it is in.

Phrases

Consider as input all of English Wikipedia stored in DFS. Goal is to build an index, on 3-grams (sequence of 3 words) that appears on exactly one page, with link to page.

Label Propagation (Graph)

Consider a large graph $G = (V, E)$ (e.g., a social network), with a subset of notes $V' \subset V$ with labels (e.g., {pos, neg}). Each node stores its label (if any) and edges.

Assign a vertex a label if (a) unlabled, (b) has ≥ 5 labeled neighbors, (c) based on majority vote.

Label Propagation (Embedding)

Consider a data set $X \subset \mathbb{R}^d$, with a subset of points $X' \subset X$ with labels (e.g., {pos, neg}). Implicitly defines graph with $V = X$ and E using $k = 20$ nearest neighbors.

Assign a vertex a label if (a) unlabled, (b) has ≥ 5 labeled neighbors, (c) based on majority vote.

Example PageRank

$$M = \begin{bmatrix} 0 & 1/2 & 0 & 0 \\ 1/3 & 0 & 1 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

Example PageRank

$$M = \begin{bmatrix} 0 & 1/2 & 0 & 0 \\ 1/3 & 0 & 1 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

Stripes:

$$M_1 = \begin{bmatrix} 0 \\ 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} \quad M_2 = \begin{bmatrix} 1/2 \\ 0 \\ 0 \\ 1/2 \end{bmatrix} \quad M_3 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad M_4 = \begin{bmatrix} 0 \\ 1/2 \\ 1/2 \\ 0 \end{bmatrix}$$

These are stored as $\big(1 : (1/3, 2), (1/3, 3), (1/3, 4)\big)$, $\big(2 : (1/2, 1)(1/2, 4)\big)$, $\big(3 : (1, 3)\big)$, and $\big(4 : (1/3, 1), (1/2, 2)\big)$.

Example PageRank

$$M = \begin{bmatrix} 0 & 1/2 & 0 & 0 \\ 1/3 & 0 & 1 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

Blocks:

$$M_{1,1} = \begin{bmatrix} 0 & 1/2 \\ 1/3 & 0 \end{bmatrix} \; M_{1,2} = \begin{bmatrix} 0 & 0 \\ 1 & 1/2 \end{bmatrix} \; M_{2,1} = \begin{bmatrix} 1/3 & 0 \\ 1/3 & 1/2 \end{bmatrix} \; M_{2,2} = \begin{bmatrix} 0 & 1/2 \\ 0 & 0 \end{bmatrix}$$

These are stored as $\big(1 : (1/2, 2)\big)$, $\big(2 : (1/3, 1)\big)$, as $\big(2 : (1, 3), (1/2, 4)\big)$, as $\big(3 : (1/3, 1)\big)$, $\big(4 : (1/3, 1), (1/2, 2)\big)$, and as $\big(3 : (1/2, 4)\big)$.