CAI: Cerca i Anàlisi d'Informació
Grau en Ciència i Enginyeria de Dades, UPC

4. Searching the Web. Pagerank

October 17, 2019

Slides by Marta Arias, José Luis Balcázar, Ramon Ferrer-i-Cancho, Ricard Gavaldà, Department of Computer Science, UPC

Contents

Searching the Web

When documents are interconnected

The World Wide Web is huge:

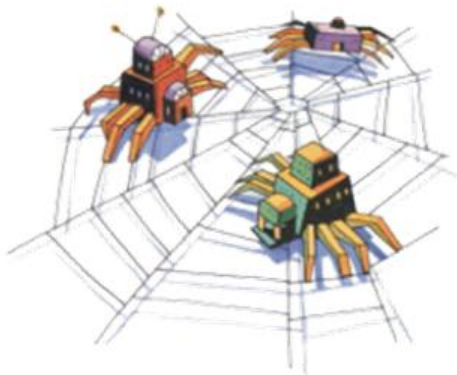
- 100,000 indexed pages in 1994
- 60,000,000,000 indexed pages in 2019

- Most queries will return *millions* of pages with high similarity.
- Content (text) alone cannot discriminate.
- Vulnerable to spam and abuse.
- Use the structure of the Web - a graph.
- Gives indications of the prestige - usefulness of each page.

Crawling

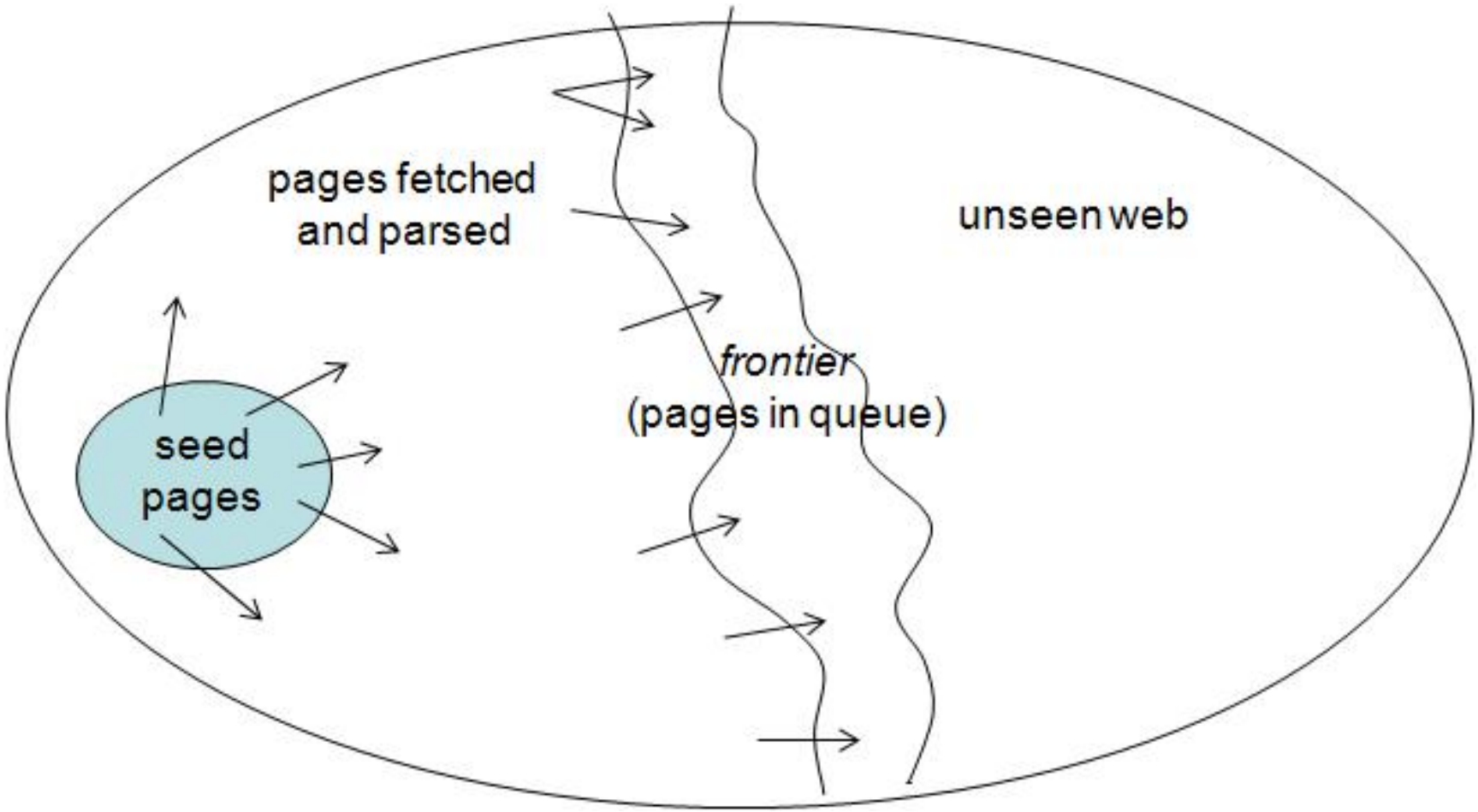
Crawler, robot, spider, wanderer . . .

Systematically explores the web & collect documents.



```
add ``seed'' URLs to queue
loop
   choose a URL from queue
   fetch page, parse it
   discard it or add it to DB
   add (new) URL's it contains to queue
end loop
```

Crawling as graph exploration



Crawling process

Exploration may be:

- breadth-first, depth-first, none of the above ...
- focused (or not): uses expressed focus or interests
  - by keywords
  - implicitly in choice of seed pages
- pages in the queue closer to focus get explored first

- Pages must be refreshed periodically.
- Pages with higher interest fetched first, refreshed more often.

The crawling process

Crawlers must be

- efficient
- robust
- polite

Crawling efficiency

- Distributed: use several machines
- Scalable: can add more machines for more throughput

- Connections have high latency
- Keep many open connections (100's?) per machine
- Try to keep all threads busy
- DNS server tends to be the bottleneck

Crawling efficiency

Some pages may be discarded:

- Duplicates
  - Fast duplicate detection a problem in itself
  - Fingerprints or k-shingles (similar to n-grams)
- Irrelevant for crawler's goals (e.g., focused crawlers)
- Unreliable or spam

Crawling robustness

- Dead URL's: *Very* common. Timeout mechanisms
- Syntactically incorrect pages
- Spider traps. Often dynamically generated
- Webspam
- Mirror sites

Crawling politeness

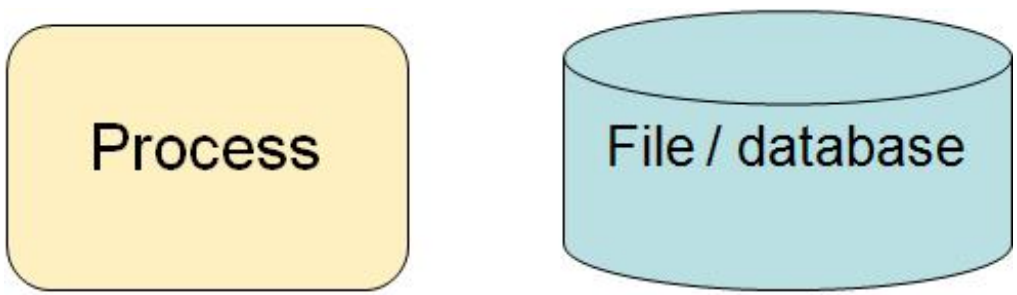
- Don't hit the same server too often, esp. downloads
- Insert wait times
- Respect robot exclusion standard
  - `/robots.txt` file: administrator preferences
  - "If you are agent X, please don't explore directory Y"

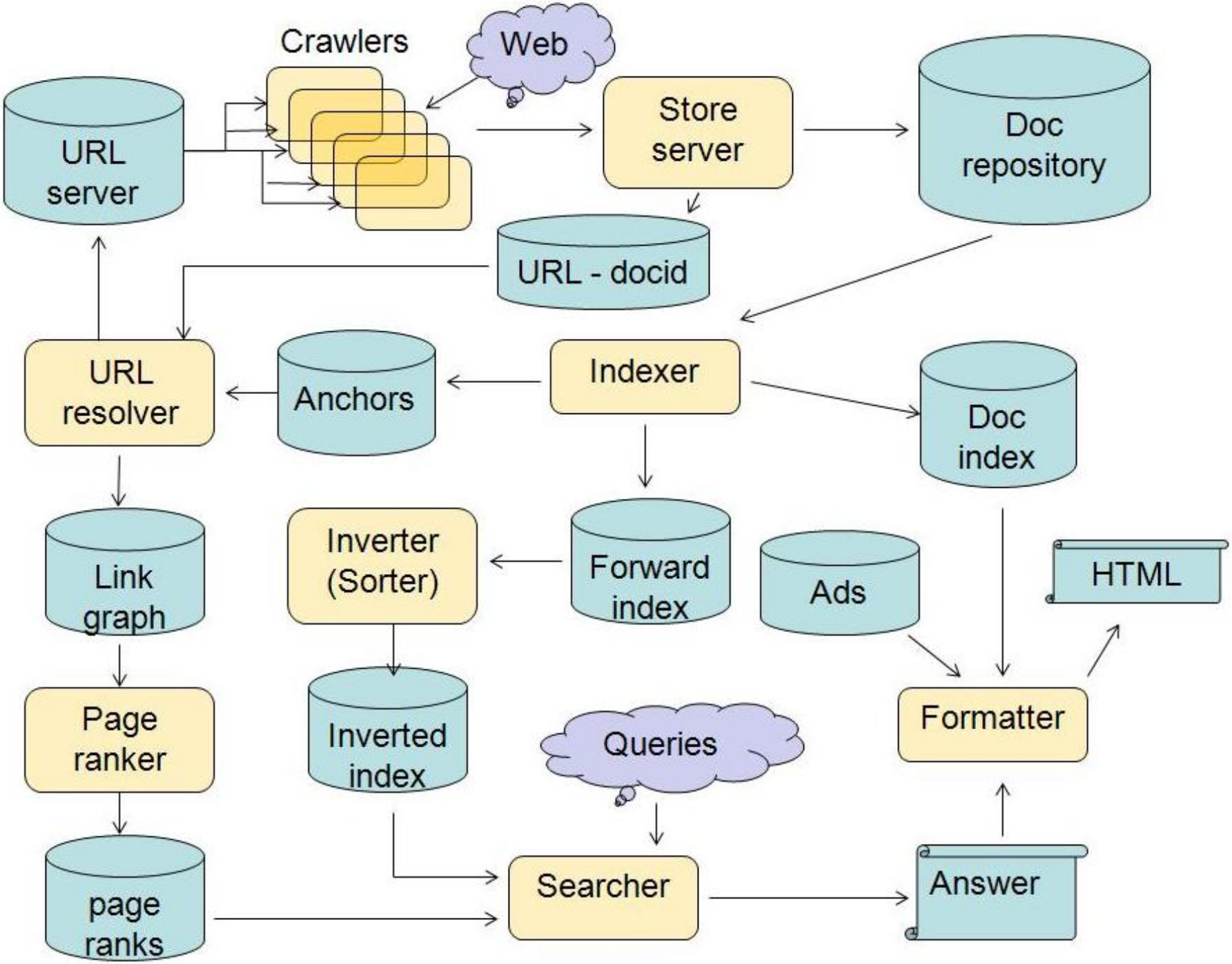
```
User-agent: *
Disallow: /cgi-bin/
Disallow: /images/
Disallow: /tmp/
Disallow: /private/
```

How Google worked in 1998

S. Brin, L. Page: “The Anatomy of a Large-Scale Hypertextual Web Search Engine”, 1998

Notation:




Crawlers
Web
URL server
Store server
Doc repository
URL - docid
URL resolver
Anchors
Indexer
Doc index
Link graph
Inverter (Sorter)
Forward index
Ads
HTML
Page ranker
Inverted index
Queries
Formatter
page ranks
Searcher
Answer

Some components

- URL store: URLs awaiting exploration
- Doc repository: full documents, zipped
- Indexer: Parses pages, separates text (to Forward Index), links (to Anchors) and essential text info (to Doc Index)
  - Text in an anchor very relevant for *target* page

    ```
    <a href="http://page">anchor</a>
    ```
  - Font, placement in page makes some terms extra relevant
- Forward index: docid $\rightarrow$ list of terms appearing in docid
- Inverted index: term $\rightarrow$ list of docid's containing term

The inverter (sorter)

Transforms forward index to inverted index

First idea:

```
for every entry document d
    for every term t in d
        add docid(d) at end of list for t;
```

Lousy locality, many disk seeks, too slow

The inverter (sorter)

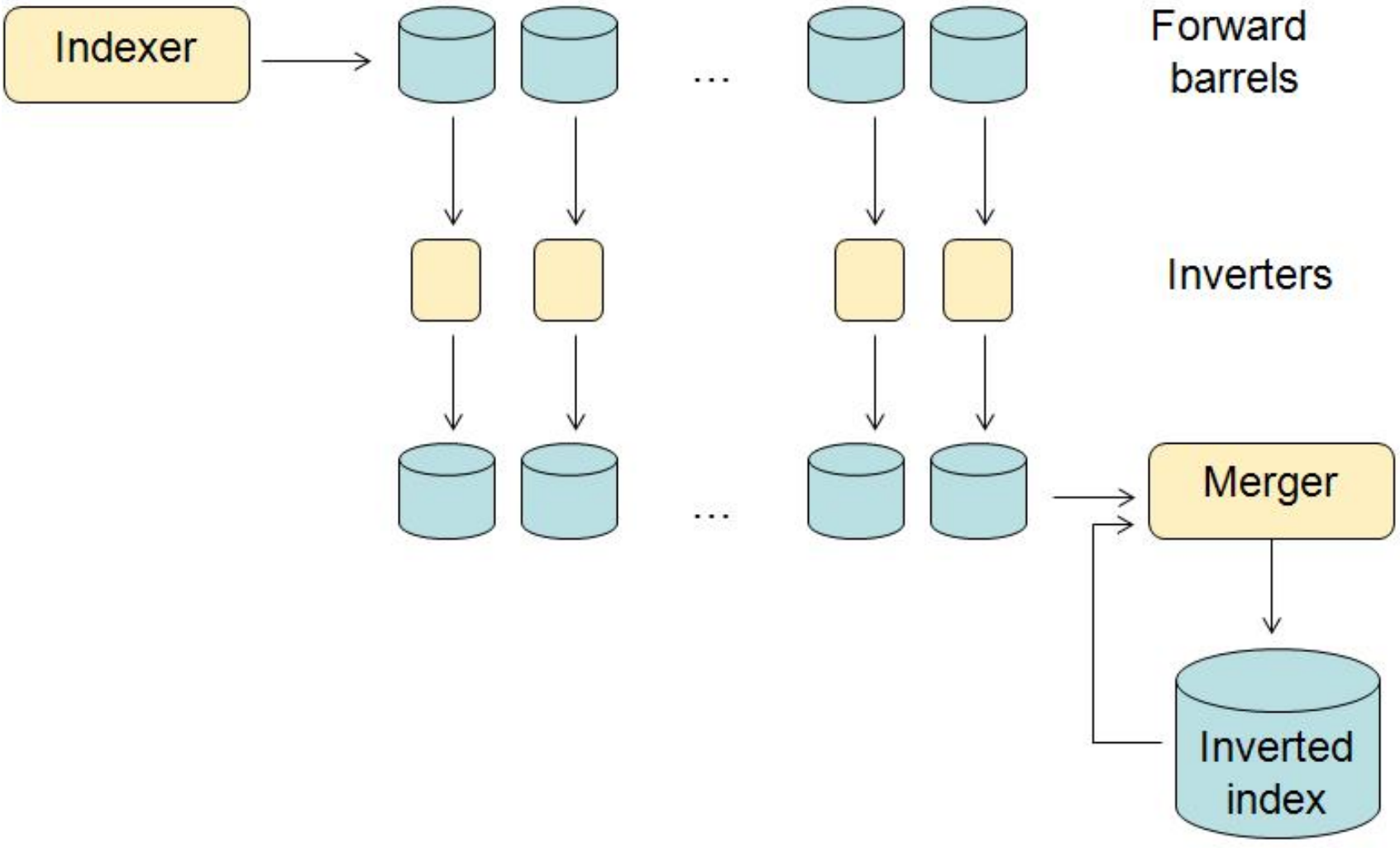
Better idea for indexing:

```
create in disk an empty inverted file, ID;
create in RAM an empty index IR;
for every document d
    for every term t in d
        add docid(d) at end of list for t in IR;
        if RAM full
            for each t, merge the list for t in IR
                into the list for t in ID;
```

Merging previously sorted lists is sequential access
Much better locality. Much fewer disk seeks.

The inverter (sorter)

The above can be done concurrently on different sets of documents:



The inverter (sorter)

- Indexer ships barrels, fragments of forward index
- Barrel size = what fits in main memory
- Separately, concurrently inverted in main memory
- Inverted barrels merged to inverted index
- 1 day instead of estimated months

Searching the Web: Meaning of Hyperlinks

When page A links to page B, this means

- A's author thinks that B's content is interesting or important or trustable
- So a link from A to B, adds to B's reputation

Inspiration for many algorithms.
Applicable to likes, follows, votes, ...

Pagerank (Brin and Page, 1998)

The idea that made Google great

But not all links give the same prestige

Intuition:

A page is important if it is pointed to by other important pages

Circular definition . . . but not a problem!

Pagerank: Definition

The web is a graph $G = (V, E)$

- $V = \{1, .., n\}$ are the nodes (that is, the pages)
- $(i, j) \in E$ if page i points to page j
- we associate to each page i, a real value p_i (i's *pagerank*)

The pagerank (prestige) of a node is passed in equal parts to the nodes to which it points.

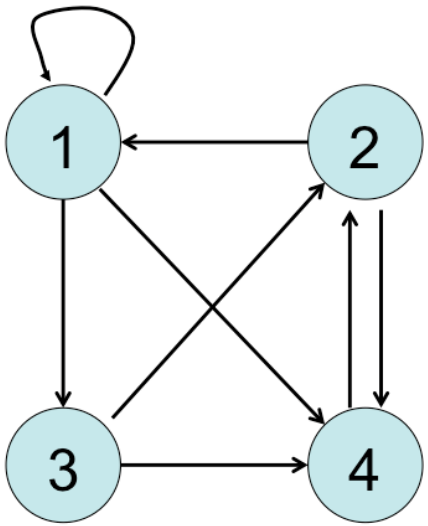
Pagerank: Definition

Definition: The vector of pageranks $(p_i)_{i \in V}$ should satisfy

1. $\sum_{i \in V} p_i = 1$
2. for all i, $p_i = \sum_{(j,i) \in E} p_j / out(j)$

$out(j)$ is the out-degree of vertex j.
All the pagerank that goes out of vertices must go into other vertices.

Pagerank, an example



$$p_i = \sum_{j \to i} \frac{p_j}{out(j)}$$

A set of $n + 1$ linear equations:

$$p_1 = \frac{p_1}{3} + \frac{p_2}{2}$$
$$p_2 = \frac{p_3}{2} + p_4$$
$$p_3 = \frac{p_1}{3}$$
$$p_4 = \frac{p_1}{3} + \frac{p_2}{2} + \frac{p_3}{2}$$
$$1 = p_1 + p_2 + p_3 + p_4$$

whose solutions is:

$p_1 = 6/23, p_2 = 8/23, p_3 = 2/23, p_4 = 7/23$

Pagerank, finding by linear algebra

Equations

- $p_i = \sum_{j:(j,i)\in E} p_j/out(j)$ for each $i \in V$
- $\sum_{i=1}^{n} p_i = 1$

where $out(i) = |\{j : (i,j) \in E\}|$ is the *outdegree* of node i

If $|V| = n$

- $n+1$ equations (but one is redundant)
- n unknowns

Could be solved, for example, using Gaussian elimination in time $O(n^3)$

Pagerank, matrix formulation

Let M be the matrix such that

- $M_{i,j} = 1/out(i)$ if $(i,j) \in E$
- $M_{i,j} = 0$ if $(i,j) \notin E$

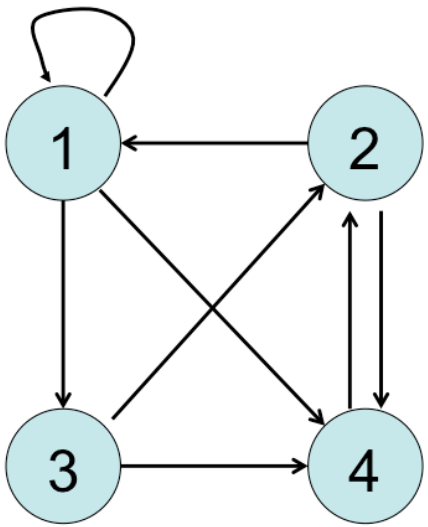
Then the system of equations above is equivalent to the matrix equation

$$M^T p = p$$

Implying: p is the (?) eigenvector of M^T associated to eigenvalue 1

Rows of M add to 1. Columns of M^T add to 1.

Pagerank, matrix formulation, example



$$M = \begin{pmatrix} \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 1 & 0 & 0 \end{pmatrix} \qquad M^T = \begin{pmatrix} \frac{1}{3} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 1 \\ \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}$$

$$\begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{pmatrix} = \begin{pmatrix} 1/3 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 1 \\ 1/3 & 0 & 0 & 0 \\ 1/3 & 1/2 & 1/2 & 0 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{pmatrix}$$

Solving $p = M^T p$ faster

$O(n^3)$ time with n = #nodes not feasible for the web size.

Power method for solving fixed point equations $x = F(x)$:

The Power Method

- Chose initial value $x(0)$ in some (unspecified) way
- Repeat $x(t) \leftarrow F(x(t-1))$
- Until convergence (i.e. $x(t) \approx x(t-1)$)

Things to prove:

- The method converges to some solution
- The method converges to a unique solution
- The method converges fast to the unique solution
- The method converges fast to the unique solution for any starting point

Solving $p = M^T p$ faster: Convergence?

In our case, F is a linear transformation given by matrix M^T:

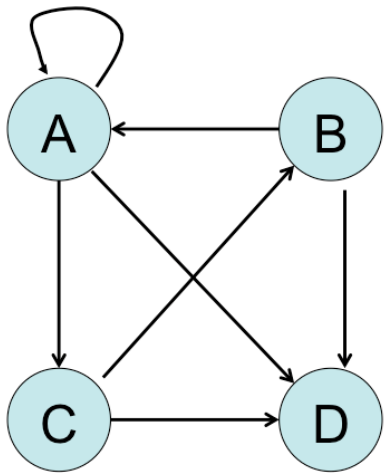
$$p(t) \leftarrow M^T p(t-1)$$

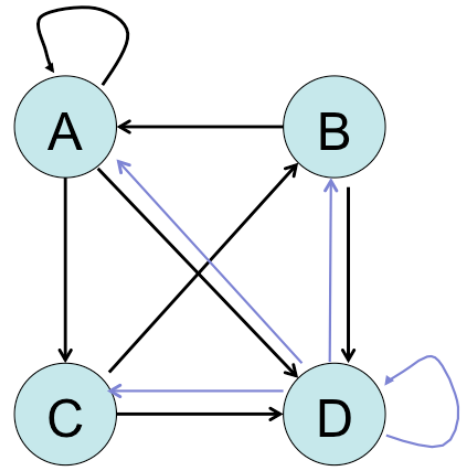
Existence, uniqueness, convergence, and speed of convergence depend on the properties of M.

Turns out that all the properties can fail for "wrong" Ms.

Pagerank: Existence

The graph on the left has no solution (check it!). but the one on the right does





Pagerank: Existence

Definition

A matrix M is stochastic, if

- All entries are in the range $[0, 1]$
- Each row adds up to 1

Theorem (Perron-Frobenius)

If M is stochastic, then it has at least one stationary vector, i.e., one non-zero vector p such that

$$M^T p = p.$$

M may not be stochastic because its rows add to $1 \ldots$ or to 0!

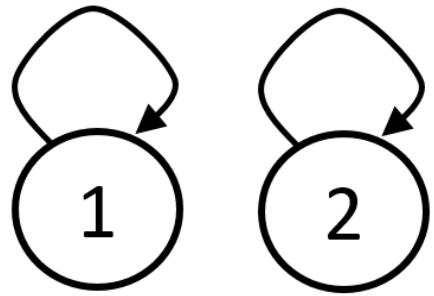
Pagerank: Existence

Fix the sum-0 rows. Saying the same in 3 ways:

- Redistribute the pagerank of a sink to all nodes.
- If $out(i) = 0$, add all edges (i, j) to E.
- If a row of M is all 0, replace it with $(1/n, \ldots, 1/n)$.

Now a solution always exists, by Perron-Frobenius.

Pagerank: Uniqueness



Infinite solutions:

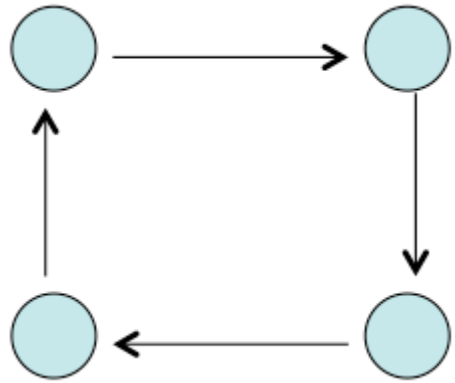
$$(1,0),\ (0,1),\ (1/2,1/2),\ (1/4,3/4),\ (7/10,3/10),\ \ldots$$

In unconnected graphs, each component retains its initial pagerank. We'll have to do something about this.

In algebra: Unconnected components have more than 1 eigenvector associated to the eigenvalue 1. If the graph is strongly connected this does not happen - multiplicity 1.

Solving $p = M^T p$ faster: Convergence?

Not necessarily



Unique solution: $(1/4, 1/4, 1/4, 1/4)$

Try initial points

- $(1, 0, 0, 0)$
- $(1/2, 0, 1/2, 0)$
- $(1/3, 2/3, 0, 0)$
- ...

Aperiodicity

Definition: Aperiodicity

A graph is aperiodic if there is no integer $k > 1$ that divides the length of every cycle.

Technicality: Extend to a matrix M in expectable way. Build $G(M)$ by putting an edge (i, j) iff $M_{i,j} \neq 0$.

Now we talk about "cycles of matrix M" by looking at $G(M)$.

When conditions are right. . .

A useful theorem from Markov Chain theory:

Theorem:

If a matrix M is strongly connected and aperiodic, then:

- M is stochastic
- Its eigenvalues satisfy $1 = \lambda_1 > \lambda_2 \geq \cdots \geq 0$
- $M^T \vec{p} = \vec{p}$ has exactly one non-zero solution such that $\sum_i p_i = 1$; call it $p^\star$
- The sequence $p(t) \leftarrow M^T p(t-1)$ satisfies

$$|p(t) - p^\star| \leq \lambda_2 \cdot |p(t-1) - p^\star|$$

- Therefore, exponential convergence to $p^\star$ from any $p(0)$:

$$|p(t) - p^\star| \leq \lambda_2^t \cdot |p(0) - p^\star|.$$

Making conditions right

1. Fix vertices with 0 outdegree as before
2. Compute M
3. Fix a damping factor $\lambda < 1$
4. Define the Pagerank matrix or Google matrix:

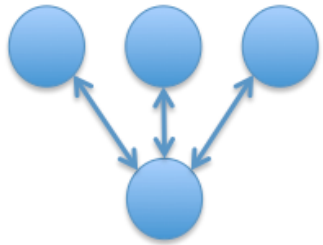
$$G = \lambda M + (1 - \lambda)\frac{1}{n}J$$

where J is the matrix containing all 1's.

Fact

G is strongly connected, aperiodic, and has 2nd eigenvalue $\lambda_2(G) \leq \lambda$

Google matrix, example



Using $\lambda = 2/3$:

$$\begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{pmatrix} = \left[\frac{2}{3} \begin{pmatrix} 0 & 1 & 1 & 1 \\ \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & 0 & 0 & 0 \end{pmatrix} + \frac{1}{3} \cdot \frac{1}{4} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \right] \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{pmatrix}$$

Exercise:

Solve this system. Trick: Use $p_2 = p_3 = p_4$.

Running time

Suppose G^T precomputed, fix tolerance ϵ

$p(0) = (1/n, \cdots, 1/n)$

repeat

$$p(t) = G^T \cdot p(t-1)$$

until $|p(t) - p(t-1)| < \epsilon$

- step 2 can be implemented in time $O(|E|)$ (often $\ll n^2$)
- number of iterations $\leq \log(|p(0) - p^\star|/\epsilon)/(1-\lambda)$

  Exercise: prove it.
- Much better than $O(n^3)$.

Observation

- λ is to ensure uniqueness and (fast) convergence. Not in pagerank definition.
- As $\lambda \to 1$, solution closer to the "true" pagerank
- As $\lambda \to 0$, solution closer to uniform (not interesting)
- As $\lambda \to 0$, faster guaranteed convergence
- Balance between speed and accuracy
- Values $0.8 \ldots 0.9$ common

Equivalently: the random surfer view

- Think of G as defining a random walk on G
- Vector $p(t)$ = probability distribution over states at time t
  - E.g., $p_i(0)$ is the probability of being at state ji at time 0
- Random surfer plays the following game:
  - Starts at node i with probability $p(0)(i)$. Then repeats forever:
  - With probability $1 - \lambda$, jump to a randomly chosen node (including itself)
  - Else, if $out(i) = 0$, jump to a randomly chosen node (including itself)
  - Else jump to any successor of i chosen at random.
- Fact: As t tends to infinity, the pagerank of i is the limit of the probability that the random surfer is at node i at time t, $p_i(t)$.

Beware of Link spam

- Link spam to increase *my* pagerank
- E.g. create a *spam farm* of pages pointing to me

- Variants of Pagerank to fight link spam.
  - TrustRank, SpamMass, ...
  - (see Leskovec, Rajaraman, Ullmann ch. 5.4)

Topic-sensitive Pagerank

Observe that pageranks are independent of user's query

- Advantages
  - Computed off-line
  - Collective reputation
- Disadvantages
  - Insensitive to particular user's needs

Topic-sensitive Pagerank

Assume there is a small set of K topics (sports, science, politics, ...)

- Each topic $k \in \{1, .., K\}$ is defined by a subset of the web pages T_k
- For each k, compute pagerank of node i for topic k:

  $p_{i,k} =$ "pagerank of node i with teleportation reduced to T_k"

- Finally compute ranking score of a page i given query q

$$score(i, q) = \sum_{k=1}^{K} sim(T_k, q) \cdot p_{i,k}$$