TDDD43

Web Information Retrieval

Fang Wei-Kleiner

ADIT/IDA
Linköping University

Outline

Early Web Search

- How to organize the Web?
  - First try: Human curated Web directories Yahoo, DMOZ.
  - Second try: Web Search
    → Information Retrieval investigates:
    Find relevant docs in a small and trusted set
    - Newspaper
    - articles, Patents
- But: Web is huge, full of untrusted documents, random things, web spam, etc.

Early Web Search Engine

- Early Web search engine worked by
  crawling the Web → terms in inverted index → query
- Ranked query processing:
  - Presence of a term in a header → higher rank
  - Large numbers of occurrences of the term → higher rank
- Term Spam

Term Spam

- A T-shirt seller could add a term MOVIE to his page, and do it thousands of times.
- When a user issued a search query with the term MOVIE, the search engine would list that page first.
- Many tricks:
  - Give it the same color as the background.
  - Go to the search engine, issue the query MOVIE → copy the 1st ranked page → using the background color to make it invisible.
- Term Spam: techniques for fooling search engines into believing your page is about something it is not.
- Term spam rendered early search engines almost useless.

PageRank

- PageRank was used to simulate where Web surfers
  - Starting at a random page
  - Would tend to congregate if they followed randomly chosen outlinks from the page at which they were currently located
  - This process were allowed to iterate many times.
  - Pages that would have a large number of surfers were considered more important than pages that would rarely be visited.
- Google prefers important pages to unimportant pages.
- Page judged not only by the terms appearing on that page, but by the terms used in or near the links to that page.
  - Spammer cannot easily get false terms added to these pages.

PageRank

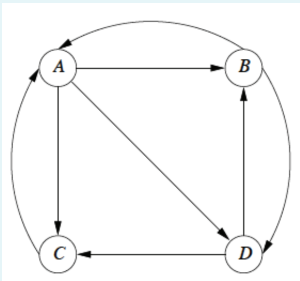
Ok, but why simulation of random surfers should allow us to approximate the intuitive notion of the importance of pages?

- Users of the Web vote with their feet.
  → They tend to place links to pages they think are good or useful pages to look at, rather than bad or useless pages.
- The behavior of a random surfer indicates which pages users of the Web are likely to visit.
  → Users are more likely to visit useful pages than useless pages.

PageRank measure has been proved empirically to work.

PageRank: Transiation Matrix

A hypothetical example of the Web



Transition matrix

$$M = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{pmatrix} 0 & 1/2 & 1 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{pmatrix} \end{array}$$

Eelement m_{ij} in row i and column j has value $1/k$ if page j has k arcs out, and one of them is to page i. Otherwise, $m_{ij} = 0$.

- Model the Web as a directed graph. Pages: nodes, Links: edges.
- The transition matrix of the Web M has n rows and columns for the Web with n pages.

PageRank: Definition

Definition (PageRank)

The probability distribution for the location of a random surfer can be described by a column vector whose jth component is the probability that the surfer is at page j. This probability is the (idealized) PageRank function.

- A random surfer at any of the n pages of the Web with equal probability. Then the initial vector v_0 will have $1/n$ for each component.
- If M is the transition matrix of the Web, then after one step, the distribution of the surfer will be Mv_0, after two steps it will be $M(Mv_0) = M^2v_0$ ...

$\rightarrow M^i v_0$ is the distribution of the surfer after i steps.

PageRank: Transiation Matrix

$$\begin{pmatrix} 0 & 1/2 & 1 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{pmatrix} = \begin{pmatrix} 9/24 \\ 5/24 \\ 5/24 \\ 5/24 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1/2 & 1 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 9/24 \\ 5/24 \\ 5/24 \\ 5/24 \end{pmatrix} = \begin{pmatrix} 15/48 \\ 11/48 \\ 11/48 \\ 11/48 \end{pmatrix}$$

...

$$\begin{pmatrix} 0 & 1/2 & 1 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 3/9 \\ 2/9 \\ 2/9 \\ 2/9 \end{pmatrix} = \begin{pmatrix} 3/9 \\ 2/9 \\ 2/9 \\ 2/9 \end{pmatrix}$$

PageRank: Definition

$$\begin{pmatrix} 0 & 1/2 & 1 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{pmatrix} = \begin{pmatrix} 9/24 \\ 5/24 \\ 5/24 \\ 5/24 \end{pmatrix}$$

The probability x_i that a random surfer will be at node i at the next step, is

$$\Sigma_j m_{ij} v_j$$

where m_{ij} is the probability that a surfer at node j will move to node i at the next step and v_j is the probability that the surfer was at node j at the previous step.

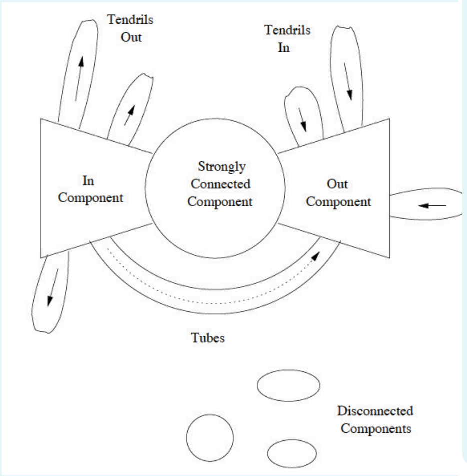
- This behavior is an example of the theory of Markov processes.

PageRank: Markov process

- It is known that the distribution of the surfer approaches a limiting distribution v that satisfies $v = Mv$, provided two conditions are met:
  - The graph is strongly connected; that is, it is possible to get from any node to any other node.
  - There are no dead ends: nodes that have no arcs out.
- Limit reached means the limiting v is an eigenvector of $M \rightarrow Mv = v$.
- M is stochastic $\rightarrow$ its columns each add up to 1.
- The principal eigenvector of M tells us where the surfer is most likely to be after a long time.
- We can compute the principal eigenvector of M by starting with the initial vector v_0 and multiplying by M some number of times, until the vector we get shows little change at each round.

Web Picture

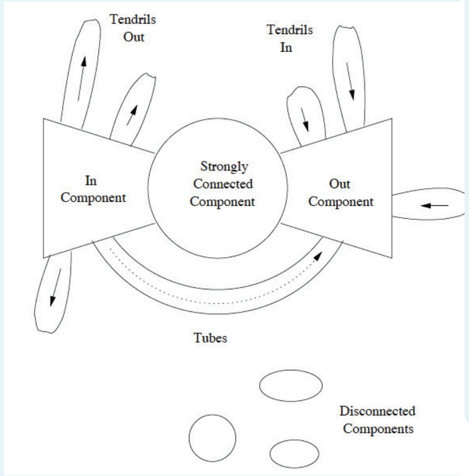
The bowtie picture of the Web



- In-component: could reach SCC, but not reachable from the SCC.
- Out-component: reachable from the SCC but unable to reach the SCC.
- Tendrils:
  - out: reachable from the in-component but not able to reach the in-component.
  - in: can reach out-component, but are not reachable from out-component.
- Tubes, isolated components

Web Picture

The bowtie picture of the Web



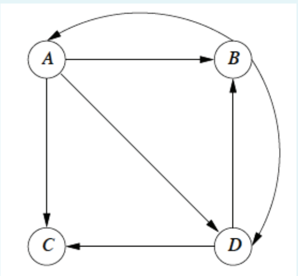
Problems:

- Violation on assumptions needed for the Markov process iteration to converge to a limit.
- Out-components: spider traps.
- Surfers starting at SCC, in-components eventually wind up in out-components or tendrils.
- Page in the SCC or in-component winds up with probability of 0.

PageRank: Dead End

With dead ends, the transition matrix of the Web is no longer stochastic $\rightarrow$ some of the columns will sum to 0 rather than 1.

Web with dead end



Transition matrix

$$M = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{pmatrix} 0 & 1/2 & 0 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{pmatrix} \end{array}$$

C is a dead end. In terms of random surfers, when surfers reaches C they disappear at the next round.

PageRank: Dead End

Starting with the vector with each component 1/4, and repeatedly multiplying the vector by M:

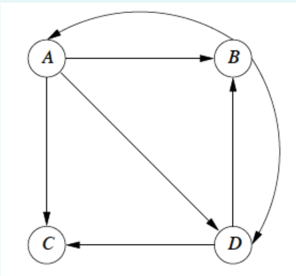
$$\begin{pmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{pmatrix} \begin{pmatrix} 3/24 \\ 5/24 \\ 5/24 \\ 5/24 \end{pmatrix} \begin{pmatrix} 5/48 \\ 7/48 \\ 7/48 \\ 7/48 \end{pmatrix} \cdots \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$\rightarrow$ After some time, all the surfers will be landing on C and drains out of the Web.

PageRank: Dead End

With dead ends, the transition matrix of the Web is no longer stochastic $\rightarrow$ some of the columns will sum to 0 rather than 1.

Web with dead end



Transition matrix

$$
\begin{array}{c c}
 & \begin{array}{cccc} A & B & C & D \end{array} \\
M = \begin{array}{c} A \\ B \\ C \\ D \end{array} & \begin{pmatrix} 0 & 1/2 & 1/4 & 0 \\ 1/3 & 0 & 1/4 & 1/2 \\ 1/3 & 0 & 1/4 & 1/2 \\ 1/3 & 1/2 & 1/4 & 0 \end{pmatrix}
\end{array}
$$

Modify the process by simulating random surfers moving about the Web.

PageRank: Modify Process for Dead End

Starting with the vector with each component 1/4, and repeatedly multiplying the vector by M:

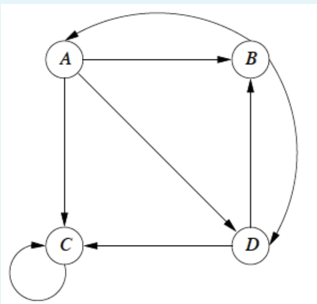
$$
\begin{pmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{pmatrix}
\begin{pmatrix} 9/48 \\ 13/48 \\ 13/48 \\ 13/48 \end{pmatrix}
\begin{pmatrix} 39/192 \\ 51/192 \\ 51/192 \\ 51/192 \end{pmatrix}
\begin{pmatrix} 153/768 \\ 205/768 \\ 205/768 \\ 205/768 \end{pmatrix}
\cdots
\begin{pmatrix} 3/15 \\ 4/15 \\ 4/15 \\ 4/15 \end{pmatrix}
$$

$\rightarrow$ Converges!

PageRank: Spider traps

A spider trap is a set of nodes with no dead ends but no arcs out.

Web with spider traps



Transition matrix

$$
\begin{array}{c c}
 & \begin{array}{cccc} A & B & C & D \end{array} \\
M = \begin{array}{c} A \\ B \\ C \\ D \end{array} & \begin{pmatrix} 0 & 1/2 & 0 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 1 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{pmatrix}
\end{array}
$$

C a simple spider trap of one node. Note that in general spider traps can have many nodes.

PageRank: Spider traps

Starting with the vector with each component 1/4, and repeatedly multiplying the vector by M:

$$
\begin{pmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{pmatrix}
\begin{pmatrix} 3/24 \\ 5/24 \\ 11/24 \\ 5/24 \end{pmatrix}
\begin{pmatrix} 5/48 \\ 7/48 \\ 29/48 \\ 7/48 \end{pmatrix}
\begin{pmatrix} 21/288 \\ 31/288 \\ 205/288 \\ 31/288 \end{pmatrix}
\cdots
\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}
$$

$\rightarrow$ All the PageRank is at C, since once there a random surfer there, he can never leave.

PageRank: Aperiodic graphs

Aperiodicity. Roughly: The pages cannot be partitioned such that the random walker visits the partitions sequentially

Graph which is not aperiodic

Transition matrix

$$M = \begin{array}{cc} & \begin{array}{cc} A & B \end{array} \\ \begin{array}{c} A \\ B \end{array} & \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{array}$$

Starting with the vector $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$, and repeatedly multiplying the vector by M:

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \dots$$

Ergodic Markov chains

- A Markov chain is ergodic iff it is irreducible and aperiodic.
- Irreducibility. Roughly: there is a path from any page to any other page.
- Aperiodicity. Roughly: The pages cannot be partitioned such that the random walker visits the partitions sequentially.

PageRank: 3 questions

$Mv = v$

- Does this converge?
  $\rightarrow$ no. As long as the graph does not fulfill those conditions. Modifying the graphs is not a good idea.
- Does it converge to what we want?
  $\rightarrow$ no. It does not really describe the random surfer's behaviour.
- Are results reasonable?
  $\rightarrow$ no. A surfer does not simply stop or get trapped repeatedly. She can always jump out and start a new page.

PageRank: Teleporting

- We modify the calculation of PageRank by allowing each random surfer a small probability of teleporting to a random page, rather than following an out-link from their current page.
- The iterative step, where we compute a new vector estimate of PageRanks v' from the current PageRank estimate v and the transition matrix M is

$$v' = \beta M v + (1 - \beta) e / n$$

PageRank: Teleporting

$$v' = \beta M v + (1-\beta)e/n$$

- β: a chosen constant, usually in the range 0.8 to 0.9.
- e: a vector of all 1's with the appropriate number of components.
- n : the number of nodes in the Web graph.
- $\beta M v$ represents the case where, with probability β, the random surfer decides to follow an out-link from their present page.
- The term $(1-\beta)e/n$ is a vector each of whose components has value $(1-\beta)/n$ and represents the introduction, with probability $(1-\beta)$, of a new random surfer at a random page.

PageRank: Teleporting

$$\text{Let } M = \begin{array}{c c} & \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \end{array} & \begin{pmatrix} 0 & 1/2 & 0 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 1 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{pmatrix} \end{array}$$

If we set β as 0.8, the equation for the iteration becomes

$$v' = \begin{pmatrix} 0 & 2/5 & 0 & 0 \\ 4/15 & 0 & 0 & 2/5 \\ 4/15 & 0 & 4/5 & 2/5 \\ 4/15 & 2/5 & 0 & 0 \end{pmatrix} v + \begin{pmatrix} 1/20 \\ 1/20 \\ 1/20 \\ 1/20 \end{pmatrix}$$

$\rightarrow$ incorporated the factor β into M by multiplying each of its elements by $4/5$.

PageRank: Teleporting

Starting with the vector with each component $1/4$, and repeatedly multiplying the vector by M:

$$\begin{pmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{pmatrix} \begin{pmatrix} 9/60 \\ 13/60 \\ 25/60 \\ 13/60 \end{pmatrix} \begin{pmatrix} 41/300 \\ 53/300 \\ 153/300 \\ 53/300 \end{pmatrix} \begin{pmatrix} 543/4500 \\ 707/4500 \\ 2543/4500 \\ 707/4500 \end{pmatrix} \cdots \begin{pmatrix} 15/148 \\ 19/148 \\ 95/148 \\ 19/148 \end{pmatrix}$$

$\rightarrow$ By being a spider trap, C has managed to get more than half of the PageRank for itself.

Ergodic Markov chains

- Theorem: For any ergodic Markov chain, there is a unique long-term visit rate for each state.
- This is the steady-state probability distribution.
- Over a long time period, we visit each state in proportion to this rate.
- It doesn't matter where we start.
- Teleporting makes the process ergodic.
- $\Rightarrow$ Web-graph+teleporting has a steady-state probability distribution.
- $\Rightarrow$ Each page in the web-graph+teleporting has a PageRank.

Topic-Specific PageRank

- Instead of generic popularity, can we measure popularity within a topic?
- Goal: Evaluate Web pages not just according to their popularity, but by how close they are to a particular topic, e.g sports or history
- Allows search queries to be answered based on interests of the user
- Example: Query Trojan wants different pages depending on whether you are interested in sports, history and computer security

Topic-Specific PageRank

- Random walker has a small probability of teleporting at any step
- Teleport can go to:
  - Standard PageRank: Any page with equal probability (To avoid dead end and spider trap problems)
  - Topic Specific PageRank: A topic specific set of relevant pages (teleport set)
- Idea: Bias the random walk
  - When walker teleports, she picks a page from a set S
  - S contains only pages that are relevant to the topic. → E.g., Open Directory (DMOZ) pages for a given topic/query
  - For each teleport set S, we get a different vector r_S

Topic-Specific PageRank

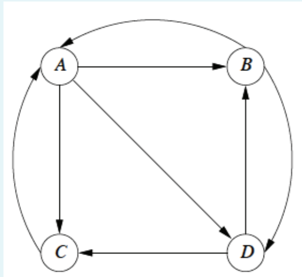
- Suppose S is a set of integers consisting of the numbers for the pages we have identified as belonging to a certain topic (called the teleport set).
- Let e_S be a vector that has 1 in the components in S and 0 in other components. Then the topic-specific PageRank for S is the limit of the iteration

$$v' = \beta M v + (1-\beta) e_S / |S|$$

where M is the transition matrix of the Web, and $|S|$ is the size of set S.

Topic-Specific PageRank

A hypothetical example of the Web



Transition matrix

$$\begin{array}{c c} & \begin{array}{cccc} A & B & C & D \end{array} \\ \mathrm{M} = \begin{array}{c} A \\ B \\ C \\ D \end{array} & \begin{pmatrix} 0 & 1/2 & 1 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{pmatrix} \end{array}$$

$$\begin{array}{c c} & \begin{array}{cccc} A & B & C & D \end{array} \\ \beta M = \begin{array}{c} A \\ B \\ C \\ D \end{array} & \begin{pmatrix} 0 & 2/5 & 4/5 & 0 \\ 4/15 & 0 & 0 & 2/5 \\ 4/15 & 0 & 0 & 2/5 \\ 4/15 & 2/5 & 0 & 0 \end{pmatrix} \end{array}$$

Where $\beta = 0.8$.

Topic Specific PageRank

Suppose our topic is represented by the teleport set $S = \{B, D\}$. Then the vector $(1-\beta)e_S/|S|$ has 1/10 for its second and fourth components and 0 for the other two components. (1/10 comes from 0.2*1/2).

$$v' = \begin{pmatrix} 0 & 2/5 & 4/5 & 0 \\ 4/15 & 0 & 0 & 2/5 \\ 4/15 & 0 & 0 & 2/5 \\ 4/15 & 2/5 & 0 & 0 \end{pmatrix} v + \begin{pmatrix} 0 \\ 1/10 \\ 0 \\ 1/10 \end{pmatrix}$$

$$\begin{pmatrix} 0/2 \\ 1/2 \\ 0/2 \\ 1/2 \end{pmatrix} \begin{pmatrix} 2/10 \\ 3/10 \\ 2/10 \\ 3/10 \end{pmatrix} \begin{pmatrix} 42/150 \\ 41/150 \\ 26/150 \\ 41/150 \end{pmatrix} \begin{pmatrix} 62/250 \\ 71/250 \\ 46/250 \\ 71/250 \end{pmatrix} \cdots \begin{pmatrix} 54/210 \\ 59/210 \\ 38/210 \\ 59/210 \end{pmatrix}$$

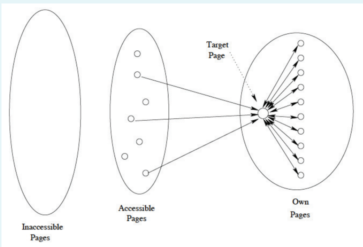
$\rightarrow$ B and D get a higher PageRank than they did before.

Link Spam

- Once Google became the dominant search engine, spammers began to work out ways to fool Google
- Spam farms were developed to concentrate PageRank on a single page
- Link spam: Creating link structures that boost PageRank of a particular page

Link Spam

The Web from the point of view of the link spammer



- A collection of pages whose purpose is to increase the PageRank of a certain page or pages is called a spam farm.
- target page t: at which spammer attempts to place as much PageRank as possible.
- A large number m of supporting pages: accumulate the portion of the PageRank that is distributed equally to all pages.

Analysis of a Spam Farm

- Taxation parameter β, typically around 0.85.
- n be pages on the Web, m be the number of supporting pages.
- x be the amount of PageRank contributed by the accessible pages.
  - $\rightarrow$ x is the sum, over all accessible pages p with a link to t, of the PageRank of p times β, divided by the number of successors of p.
- Let y be the unknown PageRank of t. We shall solve for y.

PageRank of each supporting page is $\beta y/m + (1-\beta)/n$
Then,

$$\begin{aligned} y &= x + \beta m(\beta y/m + (1-\beta)/n) + (1-\beta)/n(ignored) \\ &= x/(1-\beta^2) + c(m/n) \end{aligned}$$

where $c = \beta/(1+\beta)$.
For $\beta = 0.85$, $(1-\beta^2) = 3.6 \rightarrow$ amplified the external PageRank contribution by 360%. Increasing m will increase y.

Combating Link Spam: TrustRank

- TrustRank: topic specific PageRank with a teleport set of trusted pages. → Example: edu domains, similar domains for non US schools.
- Basic principle: while a spam page might easily be made to link to a trustworthy page, it is unlikely that a trustworthy page would link to a spam page.
- The borderline area is a site with blogs or other opportunities for spammers to create links. These pages cannot be considered trustworthy.

→ It is likely that search engines today implement this strategy routinely, so that what we think of as PageRank really is a form of TrustRank.

How hard can crawling be?

- Web search engines must crawl their documents.
- Getting the content of the documents is easier for many other IR systems.
  - E.g., indexing all files on your hard disk: just do a recursive descent on your file system
- Ok: for web IR, getting the content of the documents takes longer . . .
- . . . because of latency.
- But is that really a design/systems challenge?

Basic crawler operation

- Initialize queue with URLs of known seed pages
- Repeat
  - Take URL from queue
  - Fetch and parse page
  - Extract URLs from page
  - Add URLs to queue
- Fundamental assumption: The web is well linked.

Exercise: What's wrong with this crawler?

```
urlqueue := (some carefully selected set of seed urls)
while urlqueue is not empty:
  myurl := urlqueue.getlastanddelete()
  mypage := myurl.fetch()
  fetchedurls.add(myurl)
  newurls := mypage.extracturls()
  for myurl in newurls:
    if myurl not in fetchedurls and not in urlqueue:
      urlqueue.add(myurl)
  addtoinvertedindex(mypage)
```

What's wrong with the simple crawler

- Scale: we need to distribute.
- We can't index everything: we need to subselect. How?
- Duplicates: need to integrate duplicate detection
- Spam and spider traps: need to integrate spam detection
- Politeness: we need to be "nice" and space out all requests for a site over a longer period (hours, days)
- Freshness: we need to recrawl periodically.
  - Because of the size of the web, we can do frequent recrawls only for a small subset.
  - Again, subselection problem or prioritization

Magnitude of the crawling problem

- To fetch 20,000,000,000 pages in one month . . .
- . . . we need to fetch almost 8000 pages per second!
- Actually: many more since many of the pages we attempt to crawl will be duplicates, unfetchable, spam etc.

What a crawler must do

Be polite

- Don't hit a site too often
- Only crawl pages you are allowed to crawl: robots.txt

Be robust

- Be immune to spider traps, duplicates, very large pages, very large websites, dynamic pages etc

Robots.txt

- Protocol for giving crawlers ("robots") limited access to a website, originally from 1994
- Examples:
  - User-agent: *
    Disallow: /yoursite/temp/
  - User-agent: searchengine
    Disallow: /
- Important: cache the robots.txt file of each site we are crawling

Example of a robots.txt (nih.gov)

```
User-agent: PicoSearch/1.0
Disallow: /news/information/knight/
Disallow: /nidcd/
...
Disallow: /news/research_matters/secure/
Disallow: /od/ocpl/wag/
User-agent: *
Disallow: /news/information/knight/
Disallow: /nidcd/
...
Disallow: /news/research_matters/secure/
Disallow: /od/ocpl/wag/
Disallow: /ddir/
Disallow: /sdminutes/
```


What any crawler should do

- Be capable of distributed operation
- Be scalable: need to be able to increase crawl rate by adding more machines
- Fetch pages of higher quality first
- Continuous operation: get fresh version of already crawled pages

Resources

- Chapter 19, 20, 21 of
  Introduction to Information Retrieval
  Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze
  Ebook: *http://nlp.stanford.edu/IR-book/*
- Chapter 5 of
  Mining of Massive Datasets
  Anand Rajaraman, Jure Leskovec, Jeffrey D. Ullman
  Ebook: *http://infolab.stanford.edu/∼ullman/mmds.html*