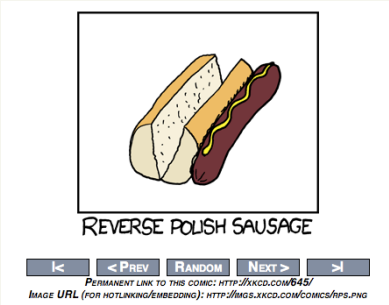


Web Crawlers and Link Analysis

David Kauchak
cs458
Fall 2011

adapted from:
http://www.stanford.edu/class/cs276/handouts/lecture15-linkanalysis.ppt
http://webcourse.cs.technion.ac.il/236522/Spring2007/ho/WCFiles/Tutorial05.ppt

Administrative

- Midterm
- Timeline
  - HW 4 due Friday (can work in pairs)
  - Assignment 4 out soon: due Friday 11/9
  - Project proposal drafts: Thursday 11/8
  - Project discussion/coordination: in-class Tuesday 11/13
    - This leaves three weeks for final projects
- Lunch Thursday

Web crawlers

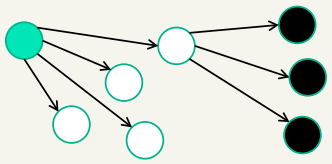
Web crawlers

Find pages on the web

How would you do it?

What are some of the challenges?

Basic crawler



Begin with "seed" URLs in the queue

- Get a URL from the queue
- Fetch the page
- Parse the page and extract URLs it points to
- Place the extracted URLs on a queue

Web crawlers

Crawling is similar at a high-level to traditional graph search

How is it different?

- Latency/bandwidth issues (we have to actually fetch each node)
- Malicious pages
  - Spam
  - Spider traps
- Politeness concerns – don't hit the same server too frequently
- Duplicate pages
- Web is not fully connected

Fetching web pages

Given a URL, we first need to fetch the actual web page

www.cs.middlebury.edu/classes/cs458/index.html

domain name | file location

What steps need to happen?

- Find the web server
  - similar to "call Dave Kauchak" – we need to know how to contact the web server
  - Computers on the web are specified by IP addresses
  - DNS (domain name service) offers a directory lookup from domain to IP address
  - DNS lookup is distributed and can be slow

Fetching web pages

Given a URL, we first need to fetch the actual web page

www.cs.middlebury.edu/classes/cs458/index.html

domain name | file location

What steps need to happen?

- Contact the web server and download the file
  - A web server is just a computer connected to the internet listening on port 80 (or sometimes 8080) for HTTP requests
  - Connect to the server and request the particular page

```
> telnet www.cs.middlebury.edu 80

GET /index.html HTTP/1.1
Host: www.cs.middlebury.edu
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)
```

Parse web page and extract URLs

Challenges/issues?

Parse web page and extract URLs

Parsing the web page

- Deal with many of the issues we talked about previously, like encoding, etc.
- Full HTML parsing can be a pain since web browsers are fault tolerant

Many HTML variants

- http://en.wikipedia.org/wiki/HTML
- Javascript, flash, …

Parse web page and extract URLs

Extract URLs

```
<b>Other information:</b>
<ul>
<li><b><a href=administrivia.html>administrivia</a></b></li>
<li><b><a href="submission">assignment submission</a></b></li>
</ul>
```

```
<a href="../midd_only/cs150solutions/set9-solution">solutions</a>
```

- Handle "relative" URLs, e.g. "administrivia.html"
- Remove duplicate URLs

Besides extracting the URLs/links for crawling purposes, is there anything else we need them for?

Connectivity Server
[*CS1:* Bhar98b, *CS2 & 3:* Rand01]

Support for fast queries on the web graph
- Which URLs point to a given URL?
- Which URLs does a given URL point to?

Stores the mappings in memory

Applications
- Crawl control
- Web graph analysis
  - Connectivity, crawl optimization
- Link analysis

Polite web crawlers

A web crawler has few constraints on which pages it can visit, but it ***must*** adhere to politeness policies

Never hit the same web server (generally IP) more frequently than once a second

Only one connection open to a giver web server at a time

robots.txt

Robots.txt

Protocol for giving spiders ("robots") limited access to a website, originally from 1994
- www.robotstxt.org/wc/norobots.html

Website announces its request on what can(not) be crawled
- For a domain, create a file `Domain/robots.txt`
- This file specifies access restrictions

Robots.txt examples

Examples

The following example "`/robots.txt`" file specifies that no robots should visit any URL starting with "`/cyberworld/map/`" or "`/tmp/`", or `/foo.html`:

```
# robots.txt for http://www.example.com/

User-agent: *
Disallow: /cyberworld/map/ # This is an infinite virtual URL space
Disallow: /tmp/ # these will soon disappear
Disallow: /foo.html
```

This example "`/robots.txt`" file specifies that no robots should visit any URL starting with "`/cyberworld/map/`" except the robot called "`cybermapper`":

```
# robots.txt for http://www.example.com/

User-agent: *
Disallow: /cyberworld/map/ # This is an infinite virtual URL space

# Cybermapper knows where to go.
User-agent: cybermapper
Disallow:
```

This example indicates that no robots should visit this site further:

```
# go away
User-agent: *
Disallow: /
```

Robots.txt example

What does this one say?

```
User-agent: *
Disallow: /yoursite/temp/
Allow: /yoursite/temp/bobs_burgers.html

User-agent: Google
Disallow:
```

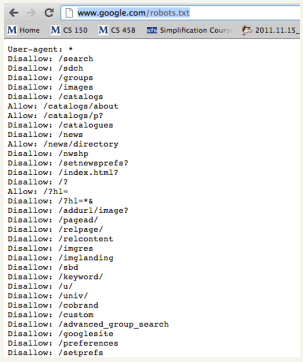
Robots.txt example

No robot should visit any URL starting with "/yoursite/temp/" except bobs_burger.html. The robot called "Google" may visit any of the pages.

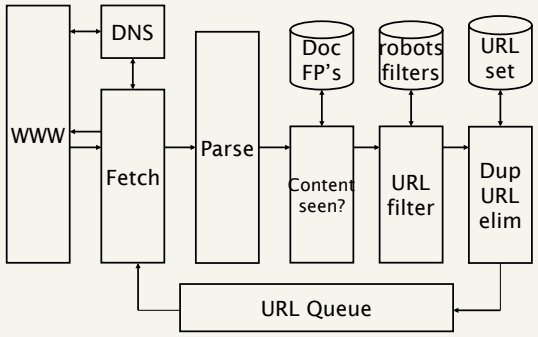
```
User-agent: *
Disallow: /yoursite/temp/
Allow: /yoursite/temp/bobs_burgers.html

User-agent: Google
Disallow:
```

not all crawlers support Allow

They can get complicated: Google.com



Basic crawl architecture



Web crawler scale

The biggest challenges for web crawlers is dealing with the size of the web

How many web pages per second would we need to download to obtain 1 billion web pages in a month?

- 30 d * 24 h * 60 m * 60 s = 2,592,000
- 1,000,000,000/2,592,000 = 385 pages/sec

Have to be multithreaded/multi-computer

Logistics become trickier

Web crawler scale issues

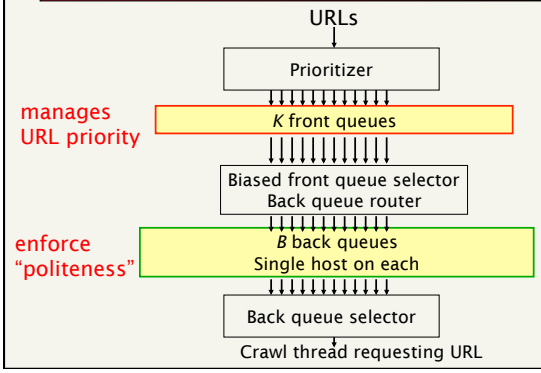
What complications does this create?

- Can't hit same web server
  - Often pages point to pages on the same server
  - Can't just wait… need to keep servers busy
  - Cache robots.txt
- Distributed computing
  - Duplicate URL detection
  - Keeping track of who's doing what
  - Cache DNS lookup since it's slow

The URL queue becomes an important data structure to try and prioritize things appropriately

- Can't just do a priority queue!

URL frontier: Mercator scheme



Priority

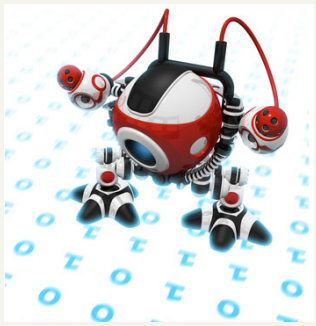
Prioritizer assigns to URL an integer priority between 1 and *K*

- Appends URL to corresponding queue

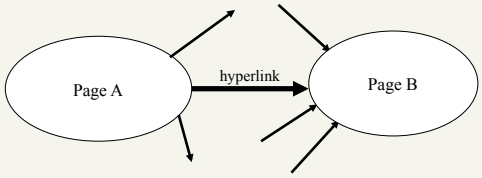
Heuristics for assigning priority?

- Refresh rate sampled from previous crawls
- Importance
- Application-specific (e.g., "crawl news sites more often")

Web crawler



The Web as a Directed Graph



A hyperlink between pages denotes author perceived relevance AND importance

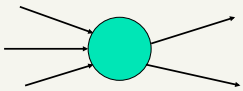
How can we use this information?

Query-independent ordering

First generation: using link counts as simple measures of popularity

Two basic suggestions:

- Undirected popularity:
  - Each page gets a score = the number of in-links plus the number of out-links (3+2=5)
- Directed popularity:
  - Score of a page = number of its in-links (3)



problems?

What is pagerank?

The random surfer model

Imagine a user surfing the web randomly using a web browser

The pagerank score of a page is the probability that a random surfing user will visit a given page

Random surfer model

We want to model the behavior of a "random" user interfacing the web through a browser

Model is independent of content (i.e. just graph structure)

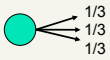
What types of behavior should we model and how?

- Where to start
- Following links on a page
- Typing in a url (bookmarks)
- What happens if we get a page with no outlinks
- Back button on browser

Random surfer model

Start at a random page

Go out of the current page along one of the links on that page, equiprobably



Random surfer model

Start at a random page

Go out of the current page along one of the links on that page, equiprobably

What about?

Random surfer model

"Teleporting"

- If a page has no outlinks always jump to random page
- With some fixed probability, randomly jump to any other page, otherwise follow links

Random surfer model

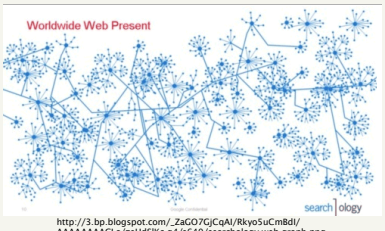
Start at a random page

If the page has no outlinks:
randomly go to another page
otherwise:
- probability α:
randomly go to another page
- probability 1-α,
Go out of the current page along one of the links on that page, equiprobably

The questions…

Given a graph and a teleporting probability, we have some probability of visiting every page

What is that probability of visiting for each page in the graph?



http://3.bp.blogspot.com/_ZaCO7GjCqAI/Rkyo5uCm8dI/AAAAAAAACLo/zsHdSlKc-q4/s640/searchology-web-graph.png

Markov process

A markov process is defined by:
- $x_1, x_2, \ldots, x_n$ a set of states
- An *n* by *n* transition matrix describing the probability of transitioning to state x_j given that you're in state x_i

$$\begin{array}{c} \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{array} \begin{array}{c} \begin{array}{cccc} x_1 & x_2 & \cdots & x_n \end{array} \\ \begin{bmatrix} & & & \\ & x_{ij}: p(x_j|x_i) & & \\ & & & \end{bmatrix} \end{array}$$

rows must sum to 1

Anybody know why it's called a *markov* process?

Markov Process

Coke vs. Pepsi Example

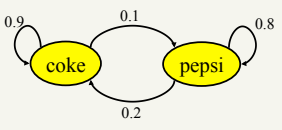
Given that a person's last cola purchase was Coke, there is a 90% chance that their next cola purchase will also be Coke.

If a person's last cola purchase was Pepsi, there is an 80% chance that their next cola purchase will also be Pepsi.

transition matrix:

	coke	pepsi
coke	0.9	0.1
pepsi	0.2	0.8

State diagram?



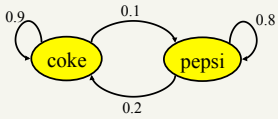
Markov Process

Coke vs. Pepsi Example (cont)

Given that a person is currently a Pepsi purchaser, what is the probability that he will purchase Coke two purchases from now?

transition matrix:

	coke	pepsi
coke	0.9	0.1
pepsi	0.2	0.8



Markov Process

Coke vs. Pepsi Example (cont)

Given that a person is currently a Pepsi purchaser, what is the probability that they will purchase Coke two purchases from now?

$$P = \begin{matrix} \text{coke} \\ \text{pepsi} \end{matrix} \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix}$$

Two scenarios:
- Pepsi -> Coke
- Coke -> Coke

Markov Process

Coke vs. Pepsi Example (cont)

Given that a person is currently a Pepsi purchaser, what is the probability that they will purchase Coke two purchases from now?

$$P = \begin{matrix} \text{coke} \\ \text{pepsi} \end{matrix} \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix}$$

Two scenarios:
- Pepsi -> Coke
  0.8 * 0.2
- Coke -> Coke
  0.2 * 0.9

Total: 0.34

Markov Process

Coke vs. Pepsi Example (cont)

Given that a person is currently a Pepsi purchaser, what is the probability that they will purchase Coke two purchases from now?

$$P = \begin{matrix} \text{coke} \\ \text{pepsi} \end{matrix} \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix} \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix} = \begin{matrix} \text{coke} \\ \text{pepsi} \end{matrix} \begin{bmatrix} 0.83 & 0.17 \\ 0.34 & 0.66 \end{bmatrix}$$

coke pepsi coke pepsi coke pepsi

Pepsi → ? ? → Coke

Markov Process

Coke vs. Pepsi Example (cont)

Given that a person is currently a Coke purchaser, what is the probability that he will purchase Pepsi **three** purchases from now?

coke pepsi coke pepsi coke pepsi

$$P^3 = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix} \begin{bmatrix} 0.83 & 0.17 \\ 0.34 & 0.66 \end{bmatrix} = \begin{bmatrix} 0.781 & 0.219 \\ 0.438 & 0.562 \end{bmatrix} \begin{matrix} \text{coke} \\ \text{pepsi} \end{matrix}$$

Steady state

In general, we can calculate the probability after *n* purchases as P^n

We might also ask the question, what is the probability of being in state coke or pepsi?

This is described by the steady state distribution of a markov process

- Note, this is a distribution over *states* not state transitions

How might we obtain this?

Steady state

We have some initial **vector** describing the probabilities of each starting state

For example, coke drinker: x = [1, 0]

coke pepsi coke pepsi coke pepsi

$$xP^3 = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 0.781 & 0.219 \\ 0.438 & 0.562 \end{bmatrix} = \begin{bmatrix} 0.781 & 0.219 \end{bmatrix}$$

We could have said a person that drinks coke 80% of the time, x = [0.80, 0.20]

$$xP^3 = \begin{bmatrix} .8 & .2 \end{bmatrix} \begin{bmatrix} 0.781 & 0.219 \\ 0.438 & 0.562 \end{bmatrix} = \begin{bmatrix} 0.712 & 0.288 \end{bmatrix}$$

Steady state

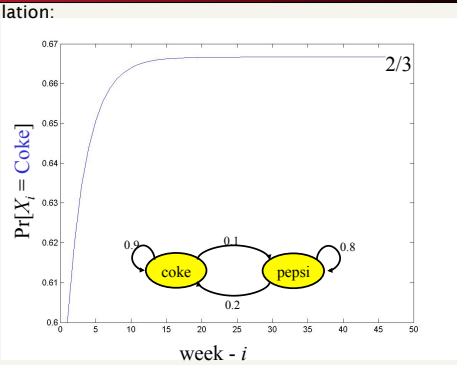
Most common:

- start with some initial x
- xP, xP^2, xP^3, xP^4, …
- For many processes, this will eventually settle

Markov Process

Coke vs. Pepsi Example (cont)

Simulation:



Back to pagerank

Can we use this to solve our random surfer problem?

- States are web pages
- Transitions matrix is the probability of transitioning to page A given at page B
- "Teleport" operation makes sure that we can get to any page from any other page

Matrix is much bigger, but same approach is taken…

- P, P^2, P^3, P^4, …

Pagerank summary

Preprocessing:

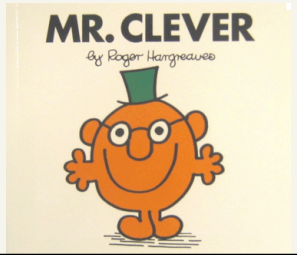
- Given a graph of links, build matrix **P**
- From it compute **steady state** of each state
- An entry is a number between 0 and 1: the pagerank of a page

Query processing:

- Retrieve pages meeting query
- Integrate pagerank score with other scoring (e.g. tf-idf)
- Rank pages by this combined score

The reality

Pagerank is used in google, but so are many other clever heuristics



Pagerank: Issues and Variants

How realistic is the random surfer model?

- Modeling the back button
- Surfer behavior sharply skewed towards short paths
- Search engines, bookmarks & directories make jumps non-random

Note that all of these just vary how we create our initial transition probability matrix

Biased surfer models

Random teleport to any page is not very reasonable

Biased Surfer Models

- Weight edge traversal probabilities based on match with topic/query (non-uniform edge selection)
- Bias jumps to pages on topic (e.g., based on personal bookmarks & categories of interest)

Topic Specific Pagerank

Conceptually, we use a random surfer who teleports, with say 10% probability, using the following rule:

- Selects a category based on a query & user-specific distribution over the categories
- Teleport to a page uniformly at random within the chosen category

What is the challenge?

Topic Specific Pagerank

Ideas?

Offline:Compute pageranks for *individual* categories

- Query independent as before
- Each page has multiple pagerank scores – one for each category, with teleportation only to that category

Online: Distribution of weights over categories computed by query context classification

- Generate a dynamic pagerank score for each page - weighted sum of category-specific pageranks

Spamming pagerank

Other link analysis

Pagerank is not the only link analysis method

- Many, many improvements/variations of pagerank
- Hubs and authorities

The **PageRank** citation ranking: bringing order to the web.
L **Page**, S Brin, R Motwani, T Winograd - 1999 - ilpubs.stanford.edu
... In Table 2, we show the resulting **page rank** percentiles for an assortment of different pages. ... Overall, our experiments with **PageRank** suggest that the structure of the Web graph is very useful for a variety of information retrieval tasks. References BP Sergey Brin and **Larry Page**. ...
Cited by 5333 Related articles All 24 versions Cite