Information Retrieval

CS 547/DS 547

Worcester Polytechnic Institute
Department of Computer Science
Instructor: Prof. Kyumin Lee

Quiz 1

- 2 Questions for 5 minutes

Project Team

- Bo Yu, Jin Yang, Kangjian Wu
- Vignesh Sundaram, Amey More, Akanksha Pawar, Padmesh Naik
- Xiaofan Zhou, Yiming Liu, Yuyuan Liu, Akhil Daphara
- Sidhant jain, Parth dhruv, Darshan Swami, Aditya Ramesh
- Chandra Rachabathuni, Ayush Shinde, Chao Wang, Anthony Chen, Adhiraj Budukh
- Joe Scheufele, Alex Alvarez, Matt Suyer, Jason Dykstra
- Adityavikram Gurao, Snehith Varma Datla, Sree Likhith Dasari, Supreeth Bandi
- Samar, Bao, Michael, Megan
- So far, 31 students formed teams.

HW1 Review

HW2

- https://canvas.wpi.edu/courses/46542

- Due date is Feb 17

Previous Class…

TF and IDF

Previous Class…

TF and IDF

tf-idf weighting

$$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \cdot \log \frac{N}{\mathrm{df}_t}$$

Summary: tf-idf

- Assign a tf-idf weight for each term t in each document *d*:

$$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \cdot \log \frac{N}{\mathrm{df}_t}$$

- The tf-idf weight . . .
  - . . . increases with the number of occurrences within a document. (term frequency)
  - . . . increases with the rarity of the term in the collection. (inverse document frequency)

Previous Class…

Cosine Similarity

Previous Class...

Cosine Similarity

BM25

Okapi BM25

$$RSV^{BM25} = \sum_{i \in q} \log \frac{N}{df_i} \times \frac{(k_1 + 1) tf_i}{k_1((1 - b) + b \frac{dl}{avdl}) + tf_i}$$

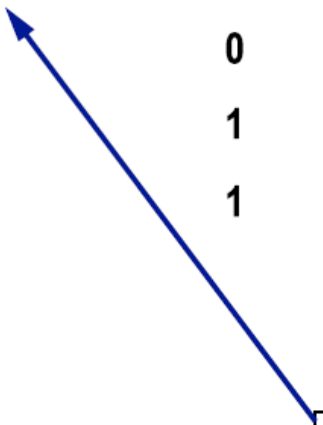
- k_1 controls term frequency scaling
  - $k_1 = 0$ is binary model; k_1 large is raw term frequency
- b controls document length normalization
  - $b = 0$ is no length normalization; $b = 1$ is relative frequency (fully scale by document length)
- Typically, k_1 is set around 1.2–2 and b around 0.75

Statistical Language Models

Three “classic” approaches to IR

Recall: Boolean Retrieval

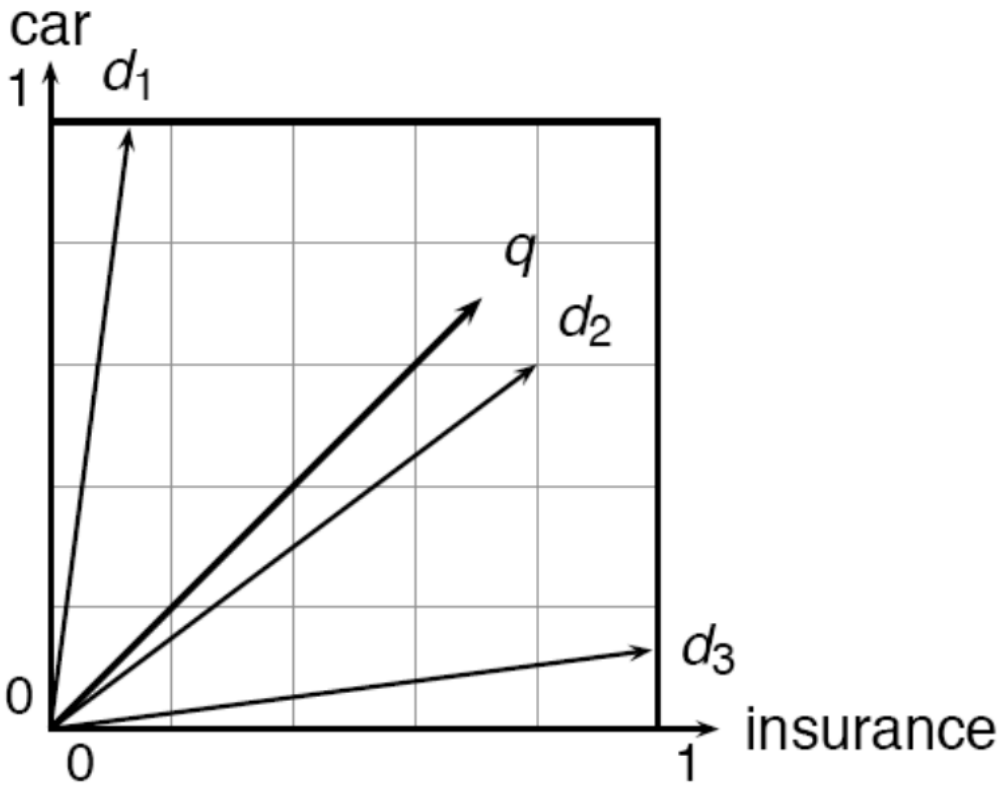
	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0



1 if play contains word, 0 otherwise

Brutus *AND* ***Caesar*** but *NOT* ***Calpurnia***

Recall: Vector Space Retrieval

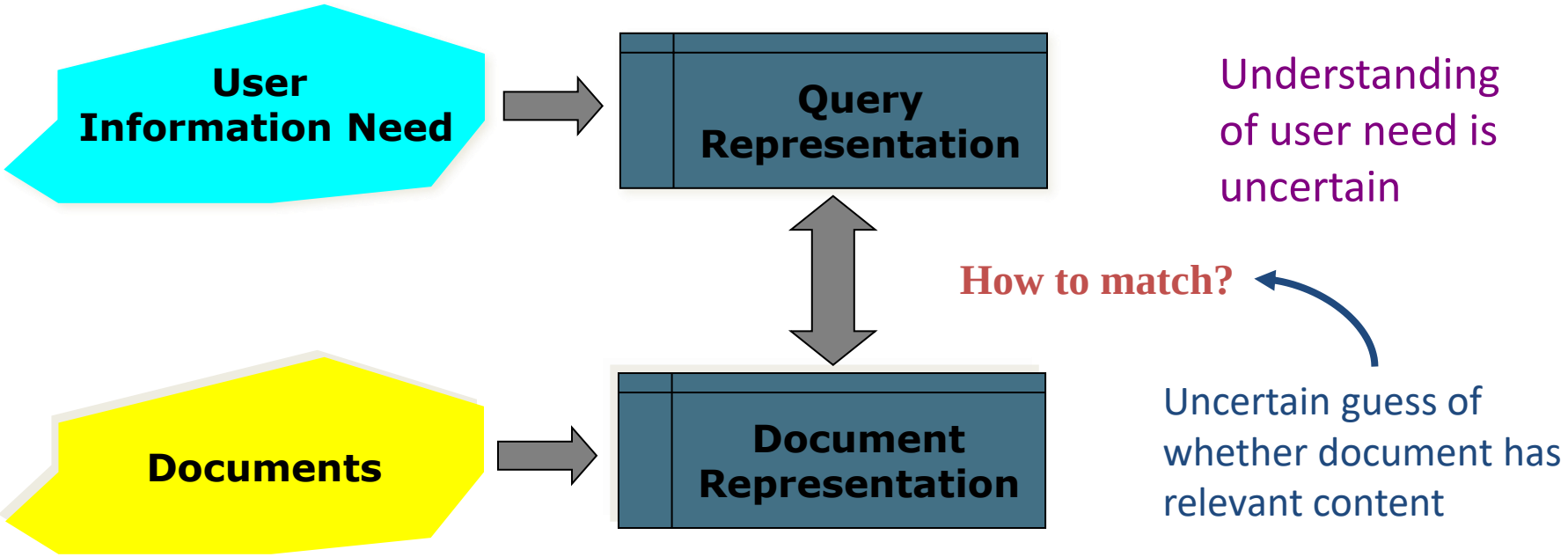


$$sim(d_j, d_k) = \frac{\vec{d}_j \cdot \vec{d}_k}{\left|\vec{d}_j\right|\left|\vec{d}_k\right|} = \frac{\sum_{i=1}^{n} w_{i,j} w_{i,k}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2} \sqrt{\sum_{i=1}^{n} w_{i,k}^2}}$$

Probabilistic IR

- Chapter 12
  - Statistical Language Models

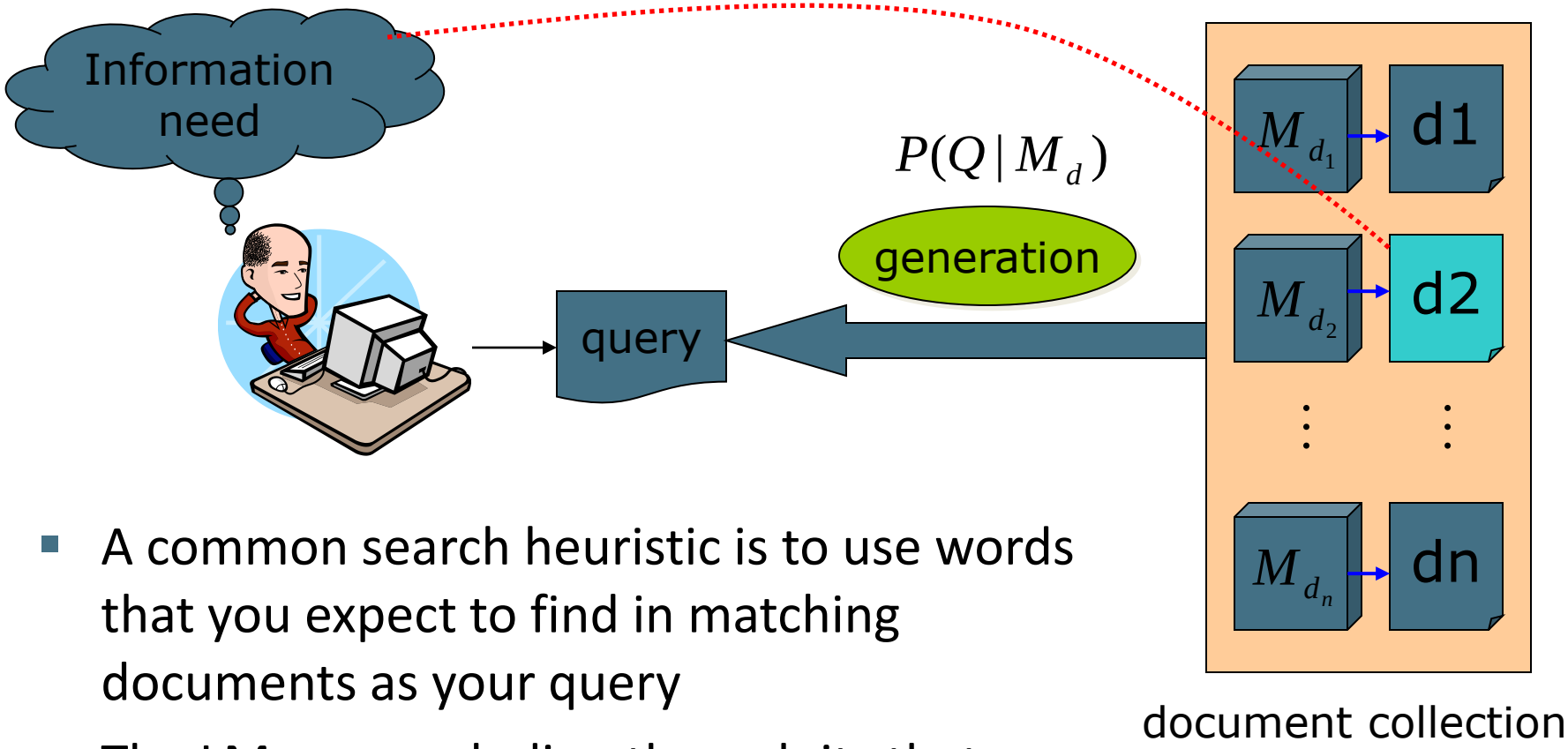
Why probabilities in IR?



In traditional IR systems, matching between each document and query is attempted in a semantically imprecise space of index terms.

Probabilities provide a principled foundation for uncertain reasoning.
Can we use probabilities to quantify our uncertainties?

IR based on Language Model (LM)



- A common search heuristic is to use words that you expect to find in matching documents as your query
- The LM approach directly exploits that idea!

What is a language model?

We can view a finite state automaton as a deterministic language model.

I wish I wish I wish I wish . . . Cannot generate: “wish I wish”

Our basic model: each document was generated by a different automaton like this except that these automata are probabilistic.

Stochastic Language Models

- Models *probability* of generating strings in the language

Model M

0.2	the
0.1	a
0.01	man
0.01	woman
0.03	said
0.02	likes
…	

the	man	likes	the	woman
0.2	0.01	0.02	0.2	0.01

multiply

P(s | M) = 0.00000008

Stochastic Language Models

- Model *probability* of generating any string

Model M1	
0.2	the
0.01	class
0.0001	sayst
0.0001	pleaseth
0.0001	yon
0.0005	maiden
0.01	woman

Model M2	
0.2	the
0.0001	class
0.03	sayst
0.02	pleaseth
0.1	yon
0.01	maiden
0.0001	woman

the	class	pleaseth	yon	maiden
0.2	0.01	0.0001	0.0001	0.0005
0.2	0.0001	0.02	0.1	0.01

$$P(s|M2) > P(s|M1)$$

Using language models in IR

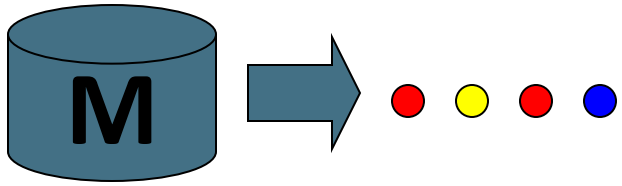
- Each document is treated as (the basis for) a language model.
- Given a query q
- Rank documents based on $P(d|q)$
- Bayes' rule

$$P(d|q) = P(q|d)P(d)/P(q)$$

- $P(q)$ is the same for all documents, so ignore
- $P(d)$ is the prior – often treated as the same for all d
  - But we can give a prior to "high-quality" documents, e.g., those with high PageRank.
- $P(q|d)$ is the probability of q given d.
- So to rank documents according to relevance to q, ranking according to $P(q|d)$ and $P(d|q)$ is equivalent.

Stochastic Language Models

- A statistical model for generating text
  - Probability distribution over a string/query in a given language



P (● ○ ● ●)

= P (●) P (○ | ●) P (● | ● ○) P (● | ● ○ ●)

Unigram and higher-order models

P (● ○ ● ●)

= P (●) P (○ | ●) P (● | ● ○) P (● | ● ○ ●)

- Unigram Language Models

  P (●) P (○) P (●) P (●)

- Bigram (generally, *n*-gram) Language Models

  P (●) P (○ | ●) P (● | ○) P (● | ●)

- We use the unigram Language Models

Where we are

- In the LM approach to IR, we attempt to model the query generation process.
- Then we rank documents by the probability that a query would be observed as a random sample from the respective document model.
- That is, we rank according to $P(q|d)$.
- Next: how do we compute $P(q|d)$?

Retrieval based on probabilistic LM

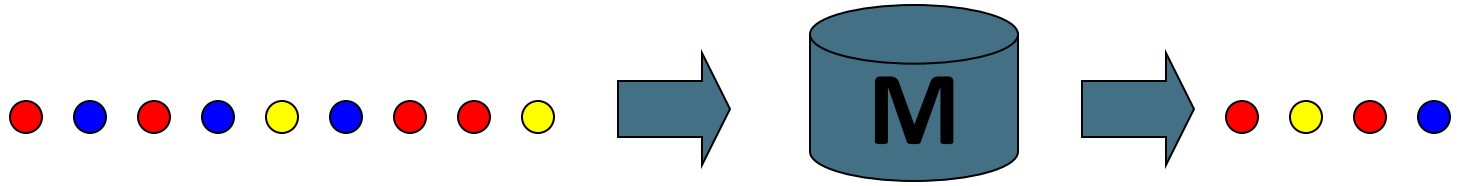
- Intuition
  - Users …
    - Have a reasonable idea of terms that are likely to occur in documents of interest.
    - They will choose query terms that distinguish these documents from others in the collection.
- Collection statistics …
  - Are integral parts of the language model.

The fundamental problem of LMs

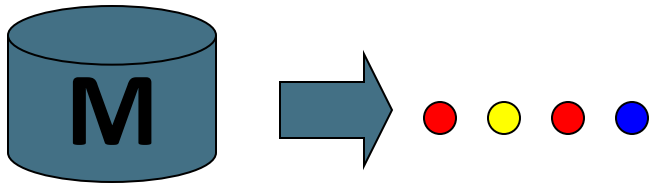
- Usually we don't know the model **M**
  - But have a sample of text representative of that model

P (● ● ● ● | M (● ● ● ● ● ● ● ● ●))

- Estimate a language model from a sample
- Then compute the observation probability

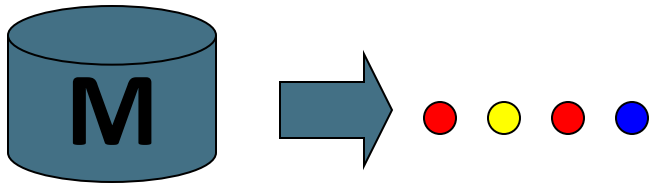


Example



- Doc 1 = “Today is a beautiful day.”
- p(today | M1) =
- p(is | M1) =
- p(a | M1) =
- p(beautiful | M1) =

Example



- Doc 2 = “Beautiful beautiful beautiful day!”
- p(today | M2) =
- p(is | M2) =
- p(a | M2) =
- p(beautiful | M2) =

Query generation probability

- Ranking formula

$$\hat{P}(q|M_d)$$

- The probability of producing the query given the language model of document *d* using *Maximum Likelihood Estimation* (MLE) is:
  - MLE means estimating a probability as the relative frequency. So this value makes the observed data maximally likely

$$\hat{P}(q|M_d) = \prod_{t \in q} \hat{P}_{\text{mle}}(t|M_d) = \prod_{t \in q} \frac{\text{tf}_{t,d}}{L_d}$$

Unigram assumption:
Given a particular language model, the query terms occur independently

M_d : language model of document d

$tf_{t,d}$: raw tf of term t in document d

L_d : total number of tokens in document d

Language Models (LMs)

- Unigram LM:
  - Bag-of-words model.
  - Multinomial distributions over words.

$$P(d) = \boxed{\frac{L_d!}{\mathrm{tf}_{t_1,d}!\mathrm{tf}_{t_2,d}!\cdots\mathrm{tf}_{t_M,d}!}} P(t_1)^{\mathrm{tf}_{t_1,d}} P(t_2)^{\mathrm{tf}_{t_2,d}} \cdots P(t_M)^{\mathrm{tf}_{t_M,d}}$$

multinomial coefficient, can leave out in practical calculations.

$$L_d = \sum_{1 \le i \le M} tf_{t_i,d}$$

The length of document d. M is the size of the vocabulary.

Query Likelihood Model

- Multinomial + Unigram:

$$P(q|M_d) = K_q \prod_{t \in V} P(t|M_d)^{\mathrm{tf}_{t,d}}$$

$$K_q = L_d!/(\mathrm{tf}_{t_1,d}!\mathrm{tf}_{t_2,d}! \cdots \mathrm{tf}_{t_M,d}!)$$

Multinomial coefficient for the query q. Can be ignored.

- Retrieve based on a language model:
  - Infer a LM for each document.
  - Estimate $P(q|M_{di})$.
  - Rank the documents according to these probabilities.

Example

$$\hat{P}(q|M_d) = \prod_{t \in q} \hat{P}_{\text{mle}}(t|M_d) = \prod_{t \in q} \frac{\text{tf}_{t,d}}{L_d}$$

- Doc 1 = “Today is a beautiful day. “
- Doc 2 = “Beautiful beautiful beautiful day!“
- Query = “today beautiful”

Insufficient data

- Zero probability $\hat{P}(t|M_d) = 0$
  - May not wish to assign a probability of zero to a document that is missing one or more of the query terms

- General approach
  - A non-occurring term is possible, but no more likely than would be expected by chance in the collection.
  - If $tf_{(t,d)} = 0$, $\hat{P}(t|M_d) \leq \mathrm{cf}_t / T$

cf_t : raw count of term t in the collection

T : raw collection size (total number of tokens in the collection)

Insufficient data

- We will use $\hat{P}(t|M_c)$ to "smooth" $P(t|d)$ away from zero.

- A simple idea that works well in practice is to use a mixture between the document multinomial and the collection multinomial distribution

Mixture model

$$\hat{P}(t|d) = \lambda \hat{P}_{\text{mle}}(t|M_d) + (1-\lambda)\hat{P}_{\text{mle}}(t|M_c)$$

- Mixes the probability from the document with the general collection frequency of the word.

- High value of λ: "conjunctive-like" search – tends to retrieve documents containing all query words (suitable for short queries)
- Low value of λ: more disjunctive, suitable for long queries

- Correctly setting λ is very important for good performance.

Basic mixture model summary

- General formulation of the LM for IR

$$P(q|d) \propto \prod_{1 \leq k \leq |q|} (\lambda P(t_k|M_d) + (1-\lambda)P(t_k|M_c))$$

individual-document model

general language model

  - The user has a document in mind, and generates the query from this document.
  - The equation represents the probability that the document that the user had in mind was in fact this one.

Example

- Document collection (2 documents)
  - d_1: Xerox reports a profit but revenue is down
  - d_2: Lucent narrows quarter loss but revenue decreases further
- Model: MLE unigram from documents; $\lambda = ½$
- Query: *revenue down*
  - $P(Q|d_1)$ =
  - $P(Q|d_2)$ =
-

$$P(q|d) \propto \prod_{1 \leq k \leq |q|} (\lambda P(t_k|M_d) + (1-\lambda)P(t_k|M_c))$$

Example

- Document collection (2 documents)
  - d_1: Xerox reports a profit but revenue is down
  - d_2: Lucent narrows quarter loss but revenue decreases further
- Model: MLE unigram from documents; $\lambda = ½$
- Query: *revenue down*
  - $P(Q|d_1) = [(1/8 + 2/16)/2] \times [(1/8 + 1/16)/2]$
    $= 1/8 \times 3/32 = 3/256$
  - $P(Q|d_2) = [(1/8 + 2/16)/2] \times [(0 + 1/16)/2]$
    $= 1/8 \times 1/32 = 1/256$
- Ranking: $d_1 > d_2$

Exercise

- Suppose, we've got 4 documents

DocID	Document text
1	click go the shears boys click click click
2	click click
3	metal here
4	metal shears click here

- Using the mixture model with lambda = 0.5, work out the per-doc probabilities for the query "click"

- collection model for “click” is
- collection model for “shears” is
- click in doc1:
- doc2:
- doc3:
- doc4:

- collection model for “click” is 7/16
- collection model for “shears” is 2/16
- click in doc1: 0.5 * 1/2 + 0.5 * 7/16 = 0.4688
- doc2: 0.7188
- doc3: 0.2188
- doc4: 0.3438

Exercise

- Suppose, we've got 4 documents

DocID	Document text
1	click go the shears boys click click click
2	click click
3	metal here
4	metal shears click here

- For the query "click shears", what's the ranking of the four documents?

click shears

- **Doc 4: 0.0645**
- Doc 1: 0.0586
- Doc 2: 0.0449
- Doc 3: 0.0137

Summary: LM

- LM approach assumes that documents and expressions of information problems are of the same type

- Computationally tractable, intuitively appealing

LMs vs. vector space model (1)

- LMs have some things in common with vector space models.
  - Term frequency is directed in the model.
    - But it is not scaled in LMs.
  - Probabilities are inherently “length-normalized”.
    - Cosine normalization does something similar for vector space.
  - Mixing document and collection frequencies has an effect similar to idf.
    - Terms rare in the general collection, but common in some documents will have a greater influence on the ranking.

LMs vs. vector space model (2)

- LMs vs. vector space model: differences
  - LMs: based on probability theory
  - Vector space: based on similarity, a geometric/ linear algebra notion
  - Collection frequency vs. document frequency
  - Details of term frequency, length normalization etc.

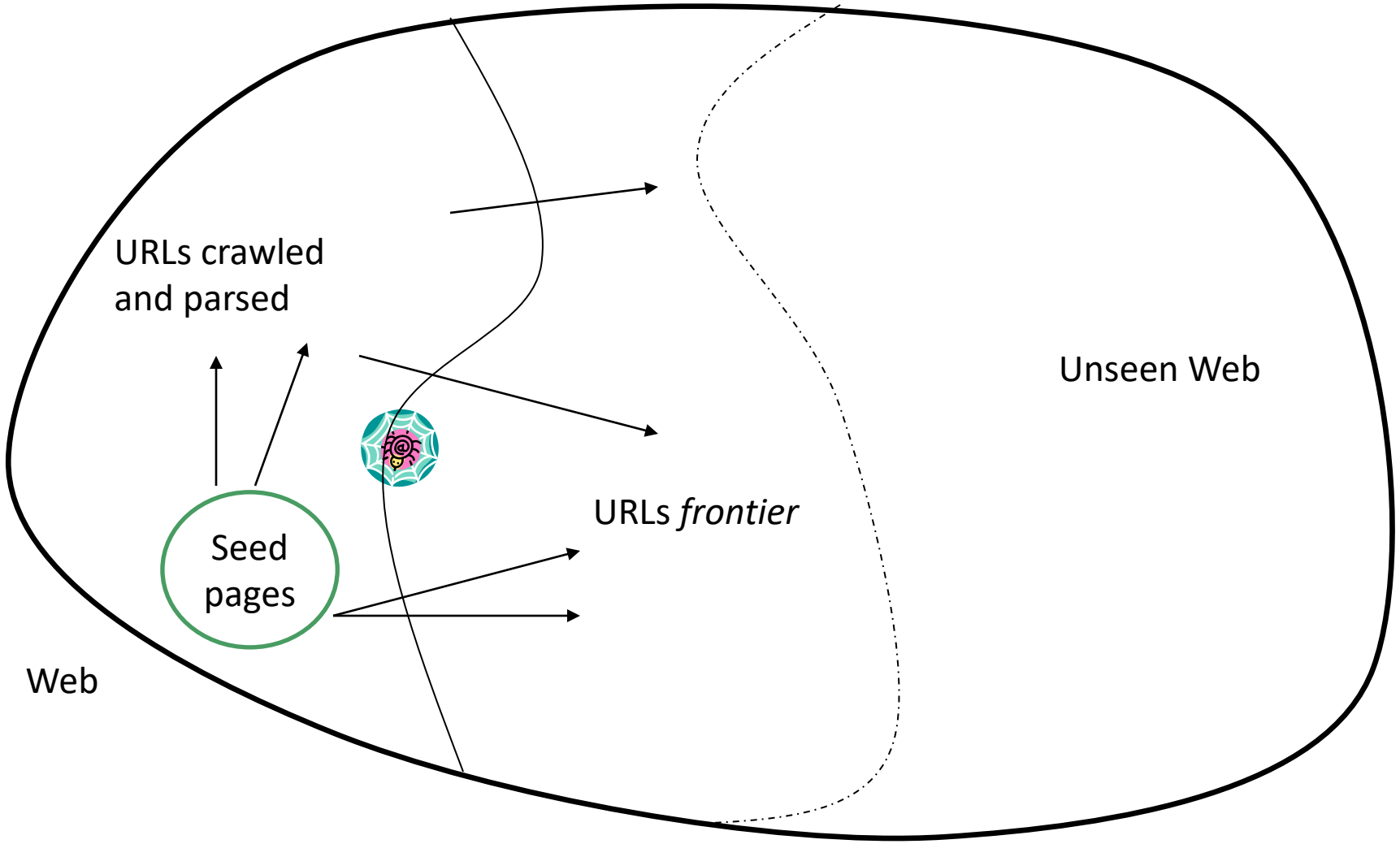
Next…

- Crawling
- Web APIs

Basic crawler operation

- Begin with known "seed" URLs
- Fetch and parse them
  - Extract URLs they point to
  - Place the extracted URLs on a queue
- Fetch each URL on the queue and repeat

Crawling picture



Simple picture – complications

- Web crawling isn't feasible with one machine
  - All of the above steps distributed
- Malicious pages
  - Spam pages
  - Spider traps – incl dynamically generated
- Even non-malicious pages pose challenges
  - Latency/bandwidth to remote servers vary
  - Webmasters' stipulations
    - How "deep" should you crawl a site's URL hierarchy?
  - Site mirrors and duplicate pages
- Politeness – don't hit a server too often

What any crawler *must* do

- Be Polite: Respect implicit and explicit politeness considerations
  - Only crawl allowed pages
  - Respect *robots.txt* (more on this shortly)
- Be Robust: Be immune to spider traps and other malicious behavior from web servers

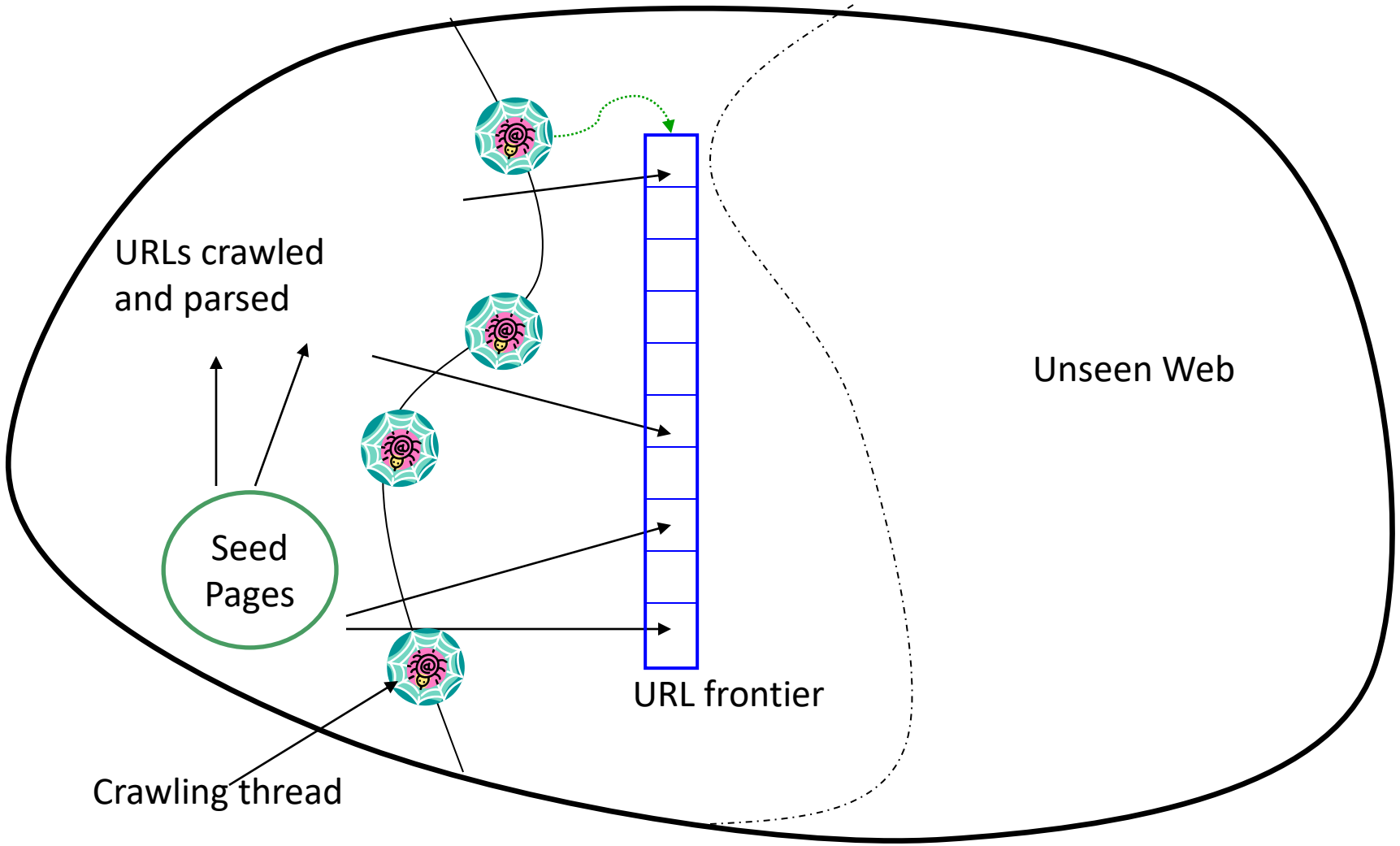
What any crawler *should* do

- Be capable of distributed operation: designed to run on multiple distributed machines
- Be scalable: designed to increase the crawl rate by adding more machines
- Performance/efficiency: permit full use of available processing and network resources

What any crawler *should* do

- Fetch pages of "higher quality" first
- Continuous operation: Continue fetching fresh copies of a previously fetched page
- Extensible: Adapt to new data formats, protocols

Updated crawling picture



URL frontier

- Can include multiple pages from the same host
- Must avoid trying to fetch them all at the same time
- Must try to keep all crawling threads busy

Explicit and implicit politeness

- Explicit politeness: specifications from webmasters on what portions of site can be crawled
  - robots.txt

- Implicit politeness: even with no specification, avoid hitting any site too often

Robots.txt

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

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

Robots.txt example

- No robot should visit any URL starting with "/yoursite/temp/", except the robot called “searchengine":

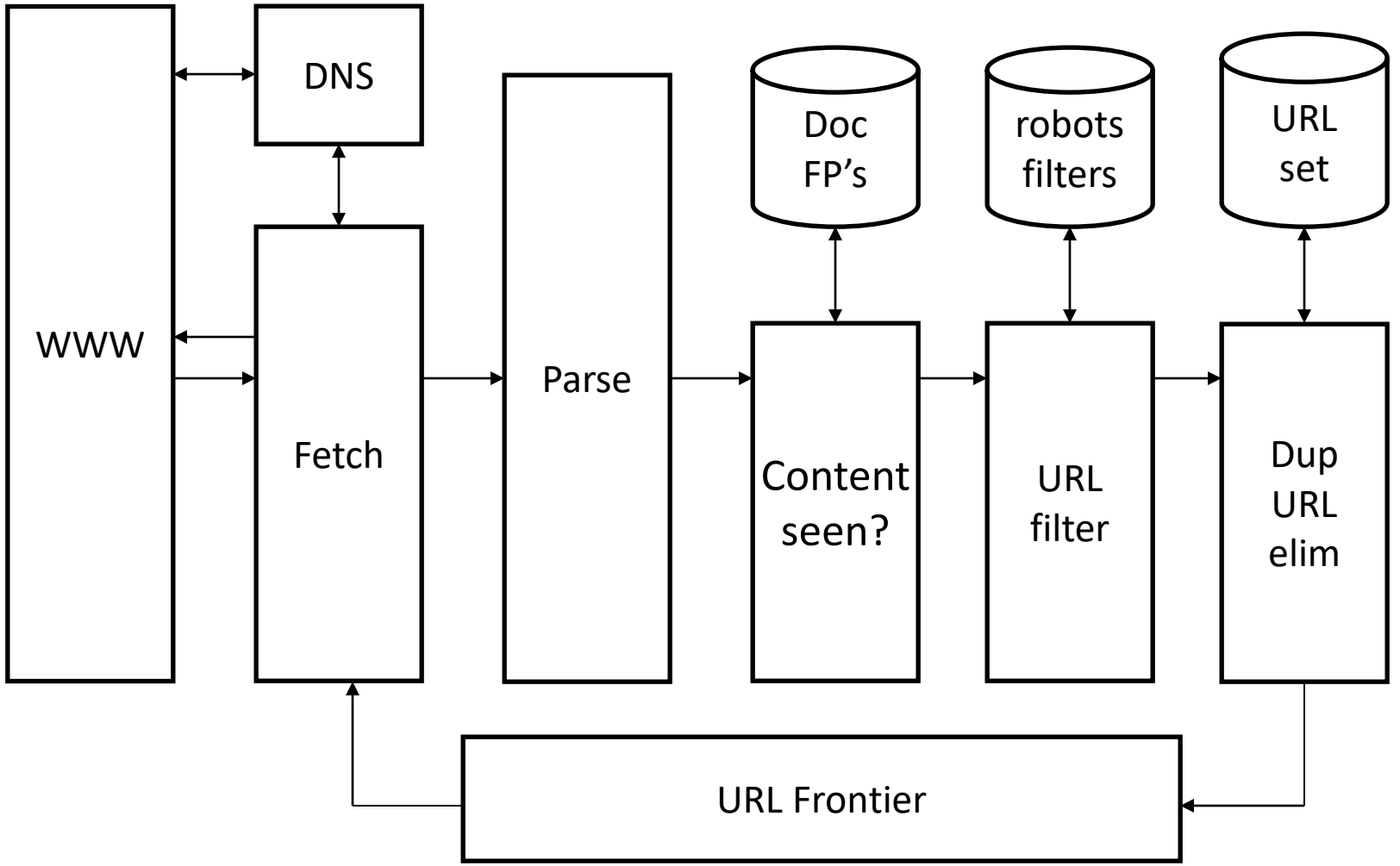
```
User-agent: *
Disallow: /yoursite/temp/

User-agent: searchengine
Disallow:
```

Processing steps in crawling

- Pick a URL from the frontier
- Fetch the document at the URL
- Parse the URL
  - Extract links from it to other docs (URLs)
- Check if URL has content already seen
  - If not, add to indexes
- For each extracted URL
  - Ensure it passes certain URL filter tests
  - Check if it is already in the frontier (duplicate URL elimination)

E.g., only crawl .edu, obey robots.txt, etc.

Basic crawl architecture



DNS (Domain Name Server)

- A lookup service on the internet
  - Given a URL, retrieve its IP address
  - Service provided by a distributed set of servers – thus, lookup latencies can be high (even seconds)
- Common OS implementations of DNS lookup are *blocking*: only one outstanding request at a time
- Solutions
  - DNS caching
  - Batch DNS resolver – collects requests and sends them out together

Parsing: URL normalization

- When a fetched document is parsed, some of the extracted links are *relative* URLs
- E.g., http://en.wikipedia.org/wiki/Main_Page has a relative link to /wiki/Wikipedia:General_disclaimer which is the same as the absolute URL http://en.wikipedia.org/wiki/Wikipedia:General_disclaimer
- During parsing, must normalize (expand) such relative URLs

Content seen?

- Duplication is widespread on the web
- If the page just fetched is already in the index, do not further process it
- This is verified using document fingerprints or shingles

Filters and robots.txt

- Filters – regular expressions for URLs to be crawled/not
- Once a robots.txt file is fetched from a site, need not fetch it repeatedly
  - Doing so burns bandwidth, hits web server
- Cache robots.txt files

Duplicate URL elimination

- For a non-continuous (one-shot) crawl, test to see if an extracted+filtered URL has already been passed to the frontier

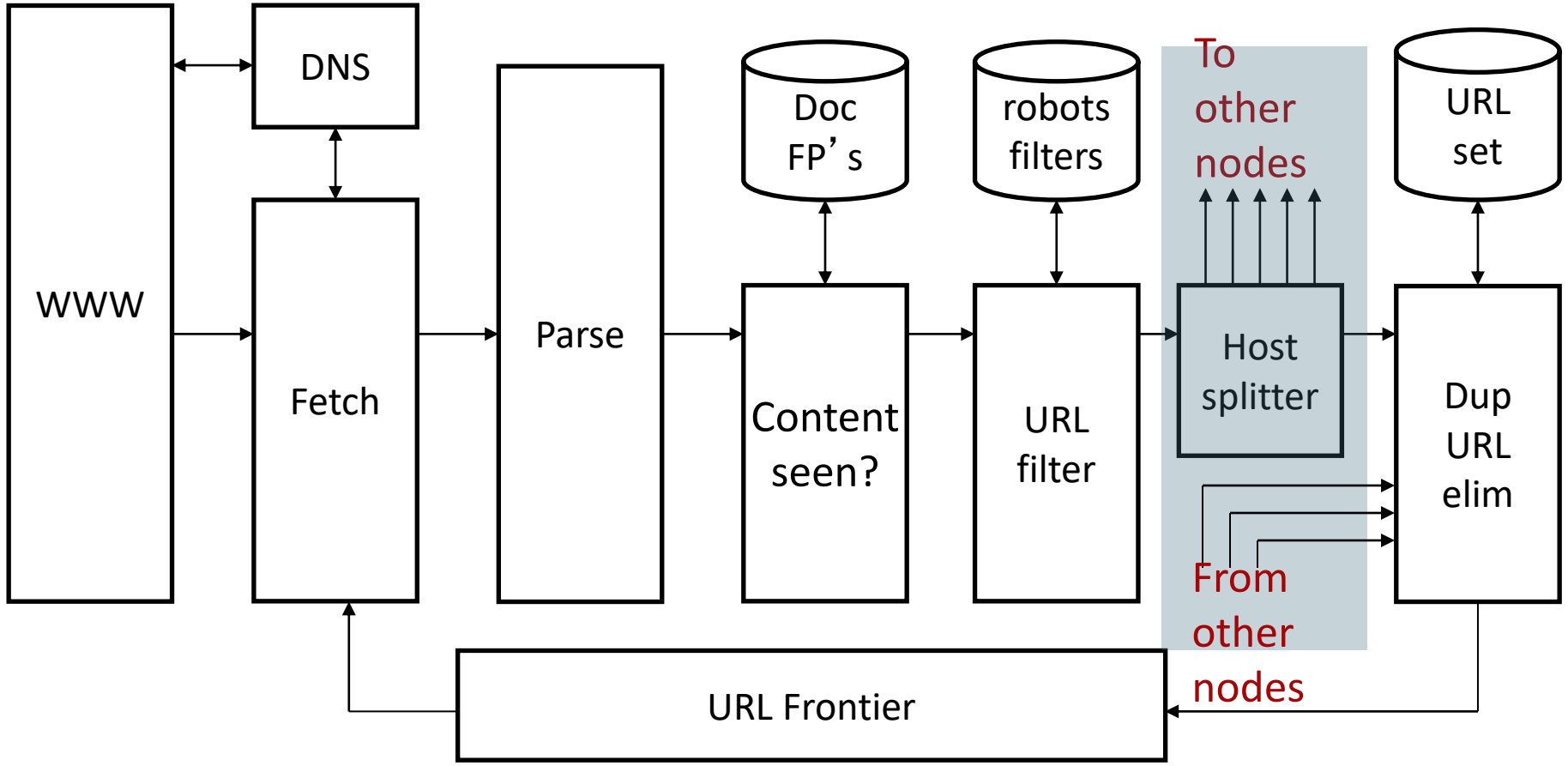
- For a continuous crawl – see details of frontier implementation

Distributing the crawler

- Run multiple crawl threads, under different processes – potentially at different nodes
  - Geographically distributed nodes
- Partition hosts being crawled into nodes
  - Hash used for partition
- How do these nodes communicate and share URLs?

Communication between nodes

- Output of the URL filter at each node is sent to the Dup URL Eliminator of the appropriate node



URL frontier: two main considerations

- Politeness: do not hit a web server too frequently
- Freshness: crawl some pages more often than others
  - E.g., pages (such as News sites) whose content changes often

These goals may conflict with each other.

(E.g., simple priority queue fails – many links out of a page go to its own site, creating a burst of accesses to that site.)

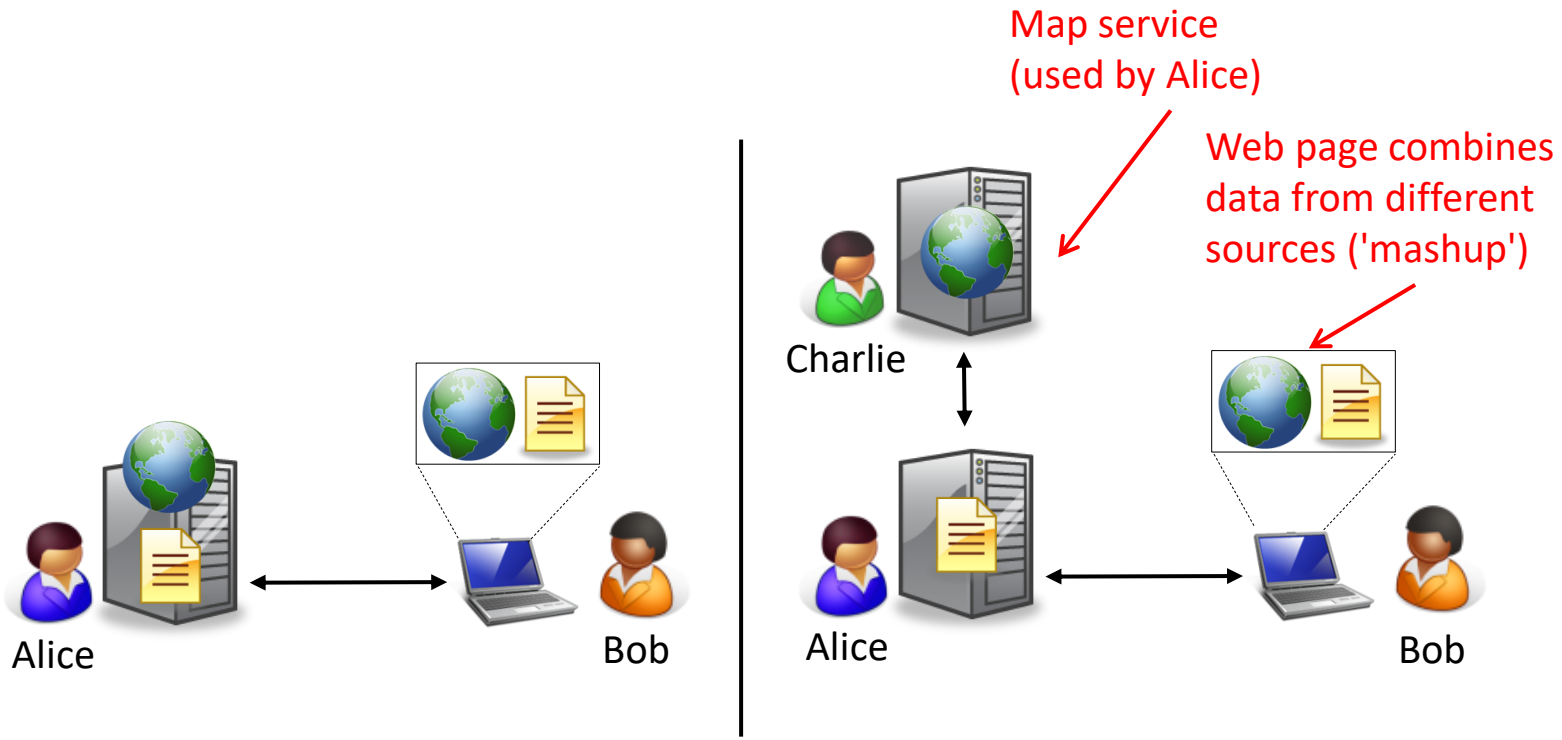
Politeness – challenges

- Even if we restrict only one thread to fetch from a host, can hit it repeatedly

- Common heuristic: insert time gap between successive requests to a host that is >> time for most recent fetch from that host

Web APIs

What is a web service?



- Intuition: An application that is accessible to other applications over the web
  - Examples: Google Maps API, Facebook Graph API, eBay APIs, Amazon Web Services APIs, ...

Available Web APIs

- Twitter: https://developer.twitter.com/en/docs
- Flickr: https://www.flickr.com/services/api/
- Google Maps: https://developers.google.com/maps/documentation
- Facebook: https://developers.facebook.com/docs/apis-and-sdks/
- Airbnb: https://www.airbnb.com/partner
- Wikipedia API: https://www.mediawiki.org/wiki/API:Main_page
- Youtube API: https://developers.google.com/youtube/v3

A list of Web APIs is in http://www.programmableweb.com/apis/directory

Play Around with Twitter API

- 1. What is Twitter?
- 2. Crawl a user’s profile
- 3. Crawl a user’s friends IDs

Calling Twitter APIs from Python

- Either directly call the API link, or use 3rd party library (e.g., Tweepy and python-twitter).
- In the sample code, register your own Twitter account, and Twitter app. Fill in the blanks in the code for the oauth authorization keys and secrets.
- https://developer.twitter.com/en/docs.html

Most APIs have Rate Limits

- Twitter has rate limits on their APIs
  - https://developer.twitter.com/en/docs/basics/rate-limits

Today: Link Analysis

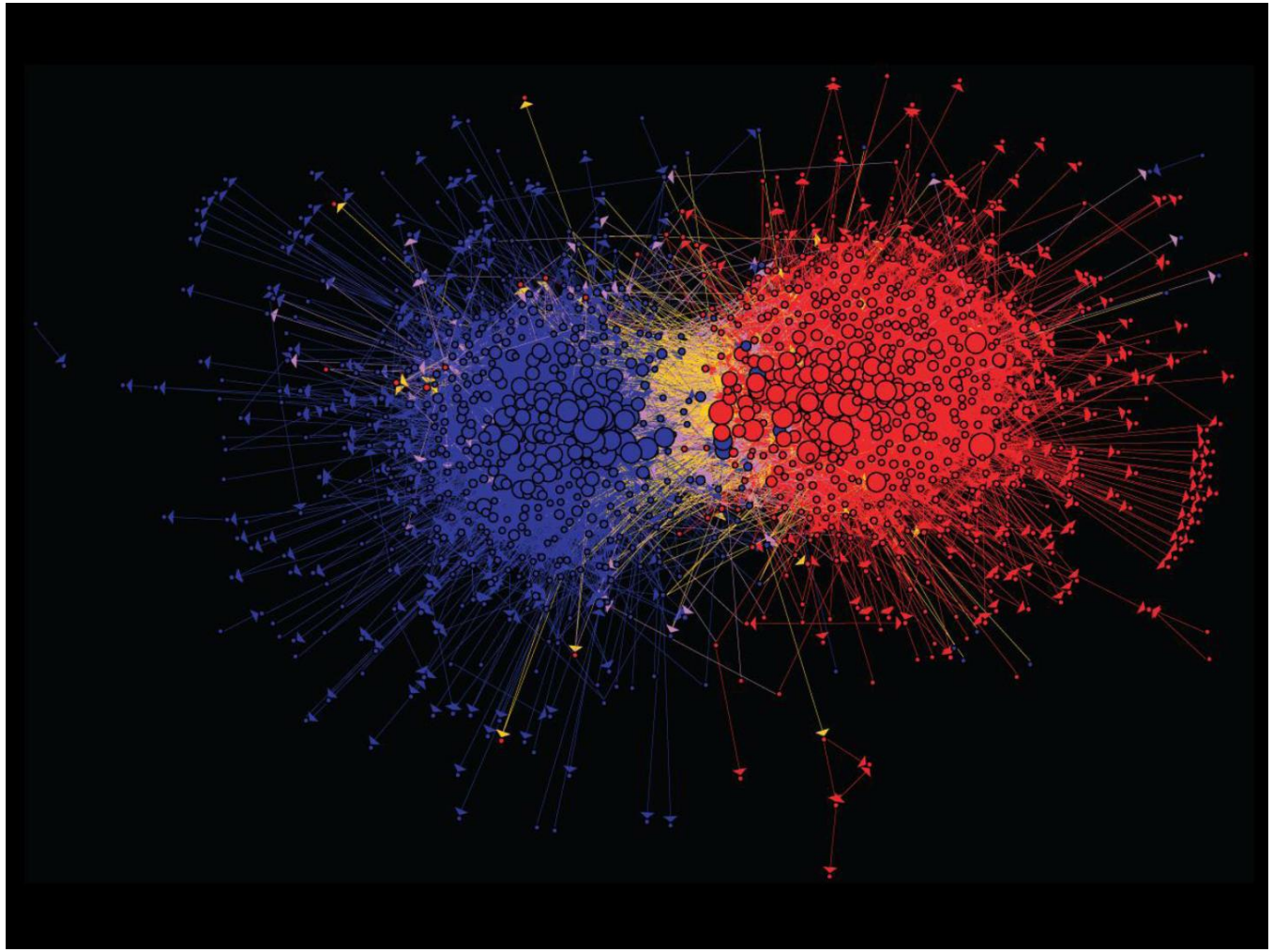
- Anchor text
- PageRank

Graph Data: Social Networks

Facebook social graph

4-degrees of separation [Backstrom-Boldi-Rosa-Ugander-Vigna, 2011]

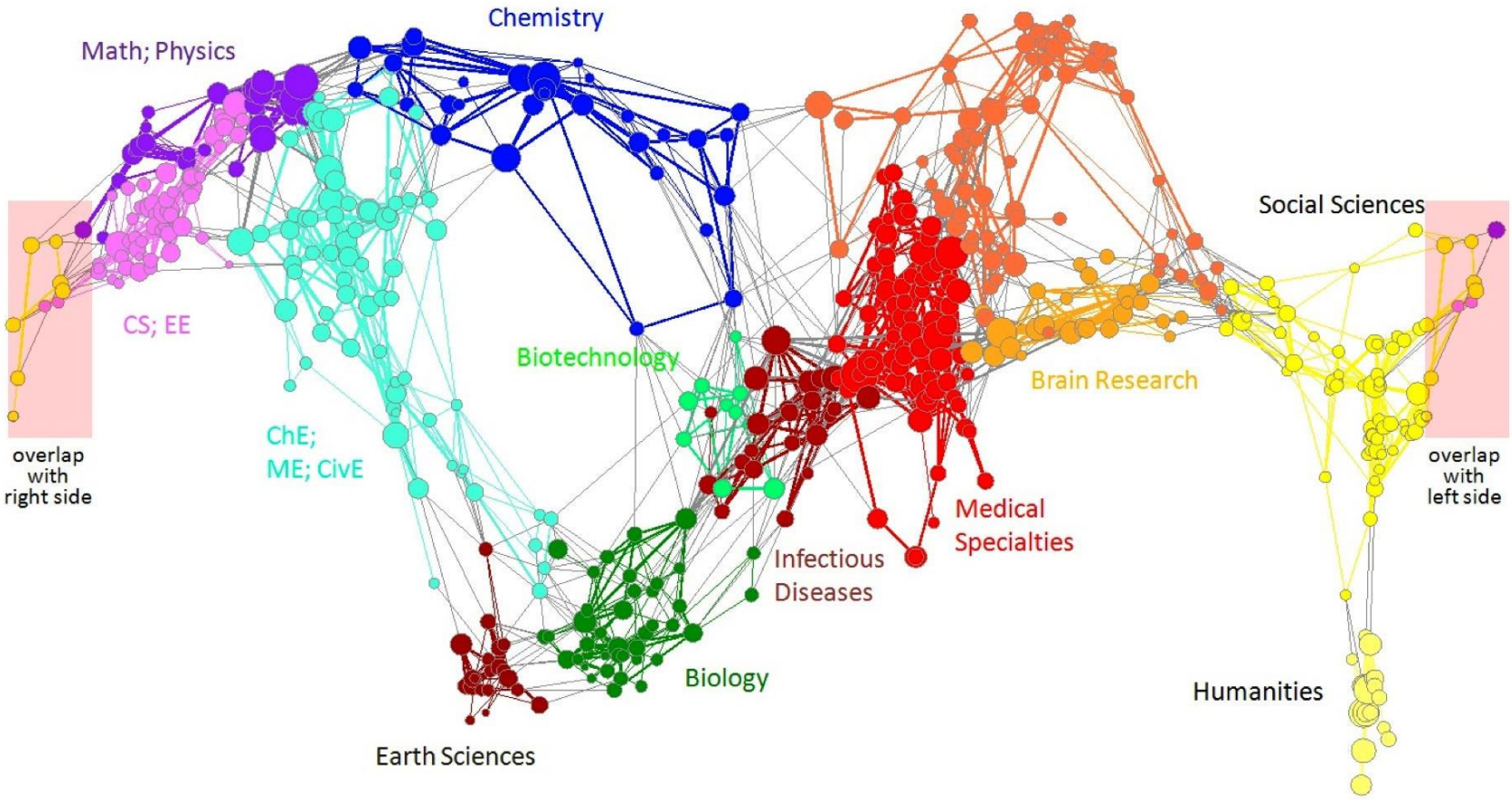
Graph Data: Media Networks



Connections between political blogs

Polarization of the network [Adamic-Glance, 2005]

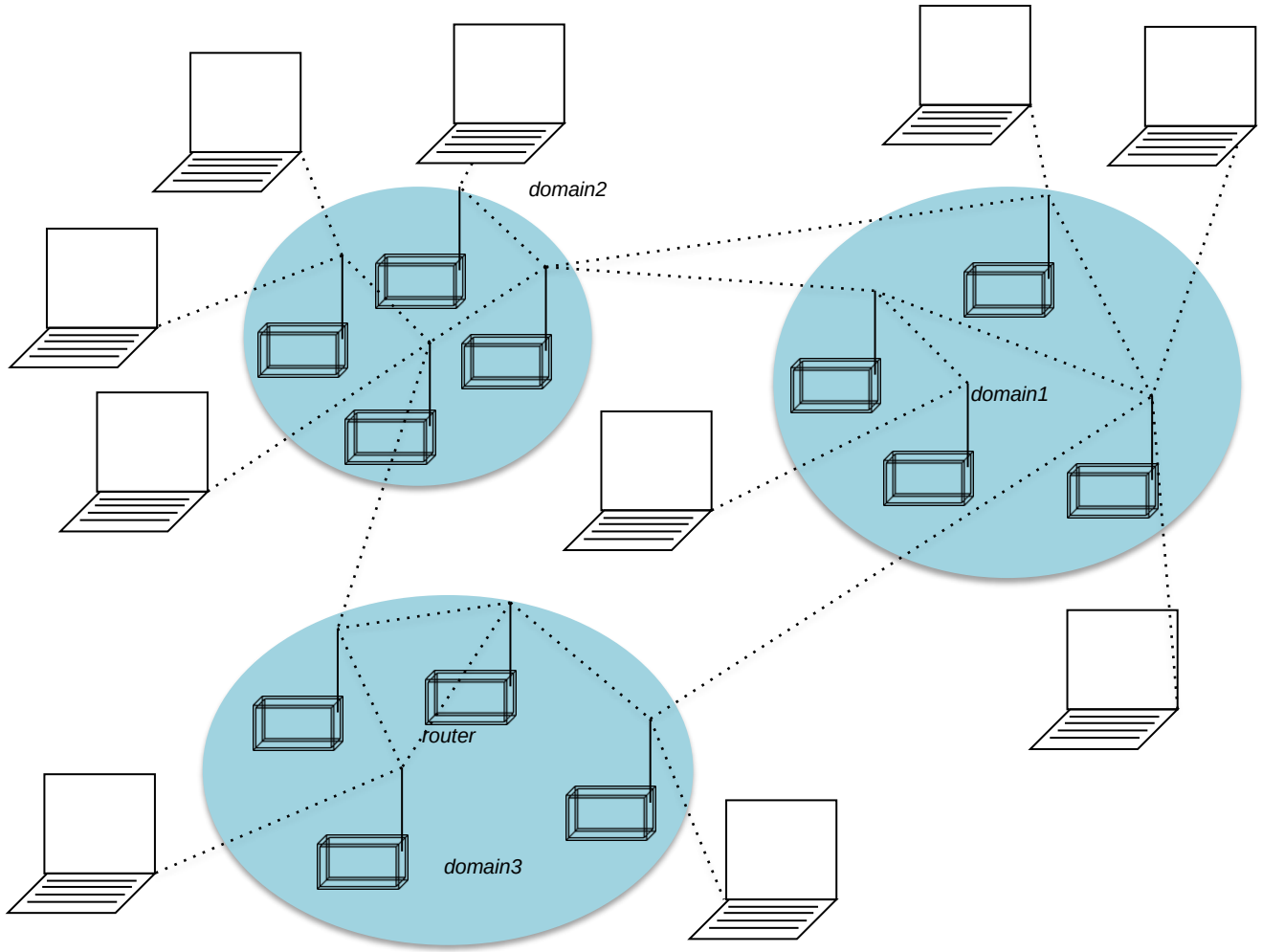
Graph Data: Information Nets



Citation networks and Maps of science

[Börner et al., 2012]

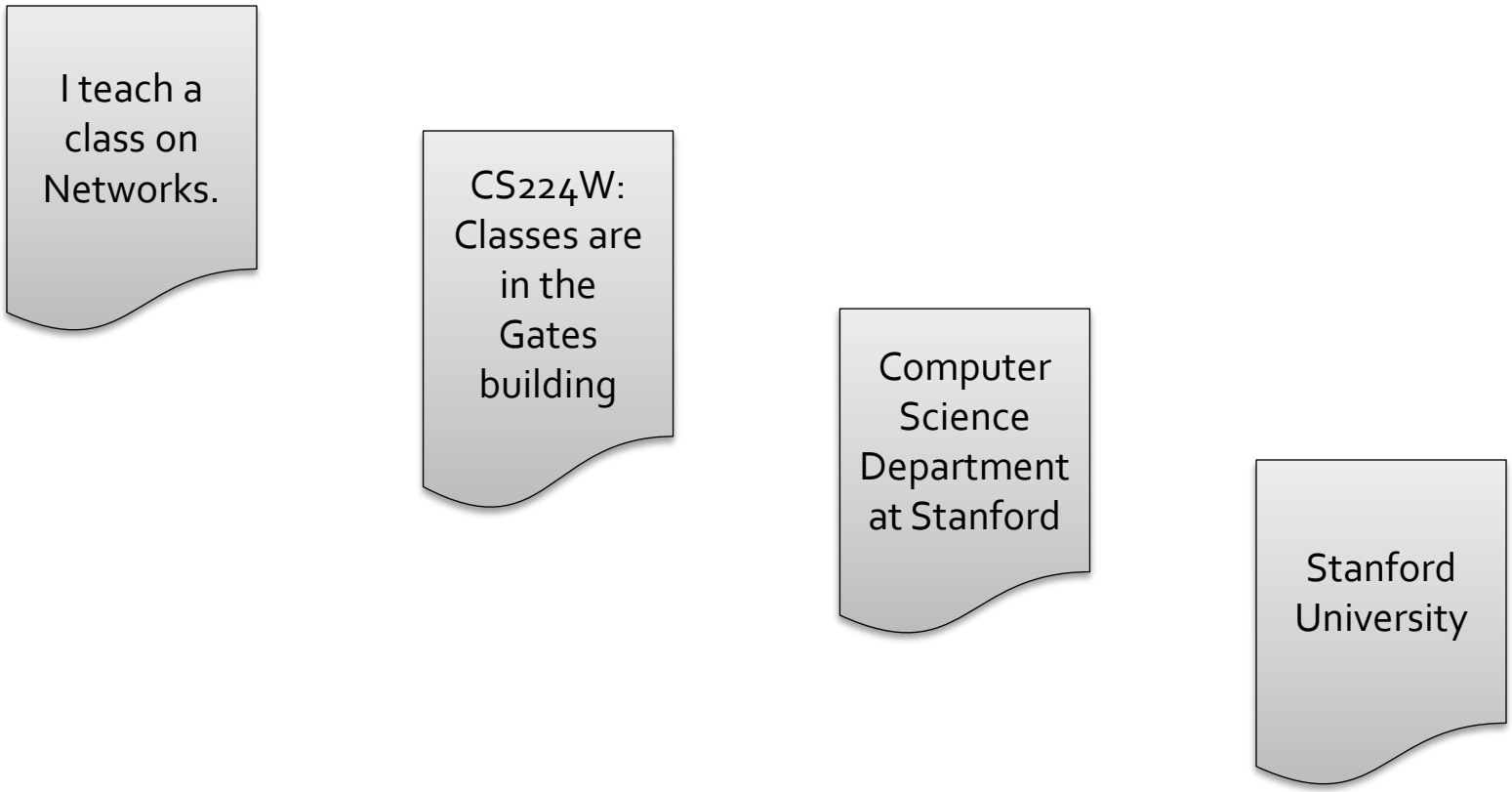
Graph Data: Communication Nets



Internet

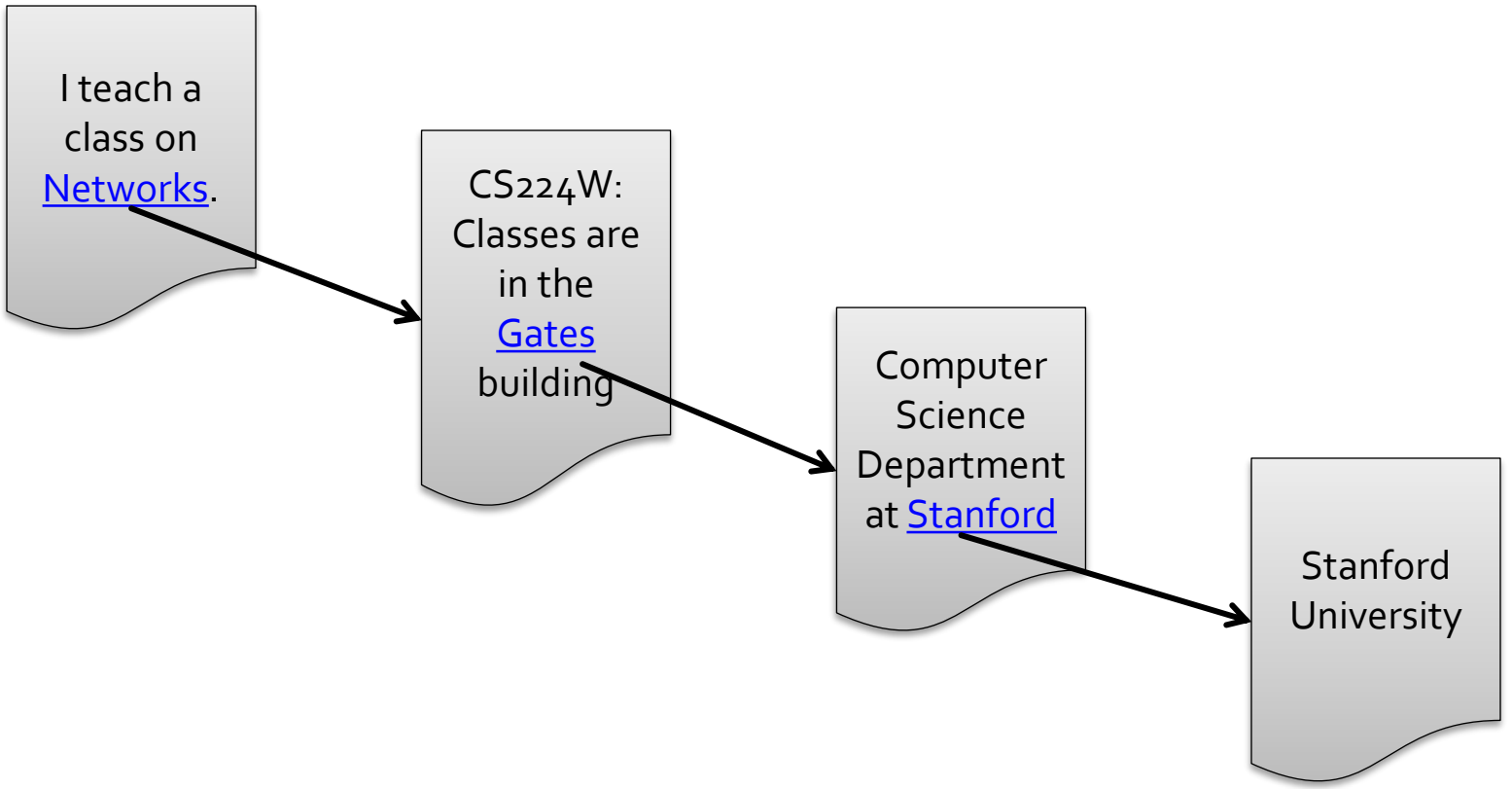
Web as a Graph

- **Web as a directed graph:**
  - **Nodes: Webpages**
  - **Edges: Hyperlinks**

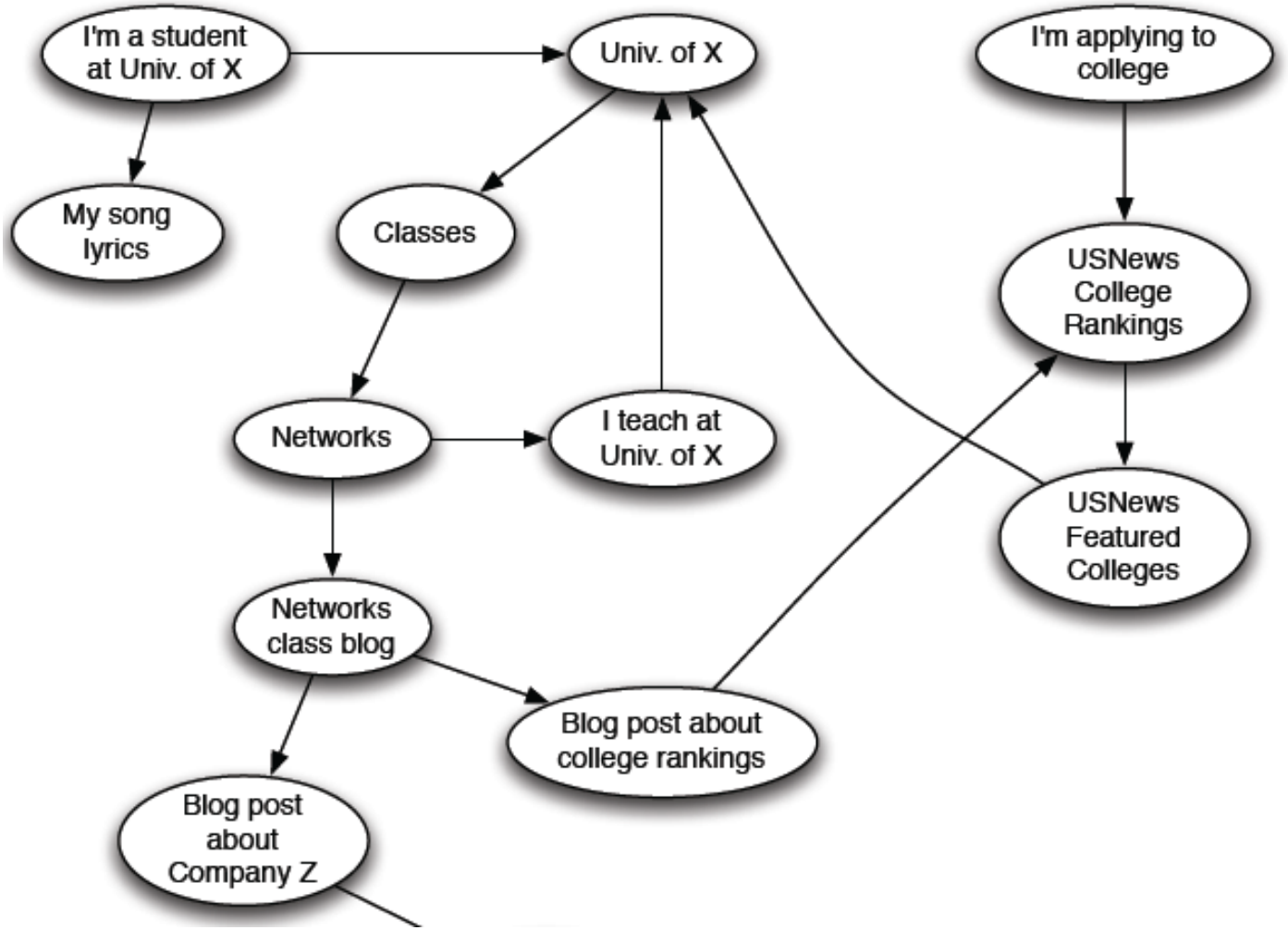


Web as a Graph

- **Web as a directed graph:**
  - **Nodes: Webpages**
  - **Edges: Hyperlinks**



Web as a Directed Graph

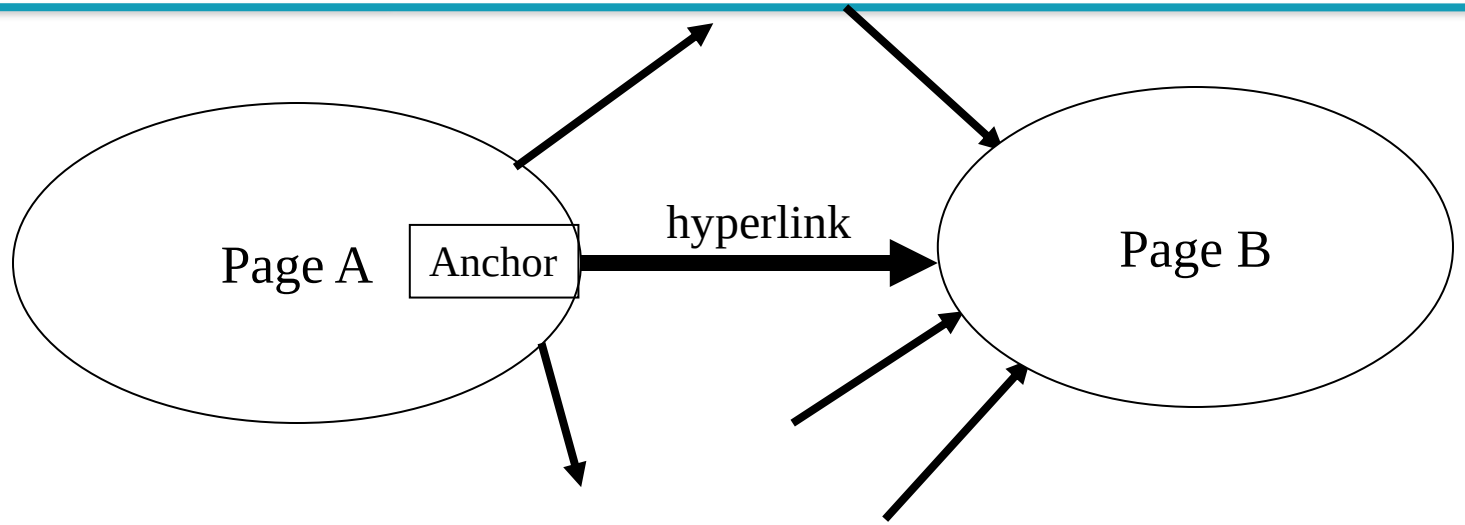


Broad Question

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

Anchor Text

The Web as a Directed Graph

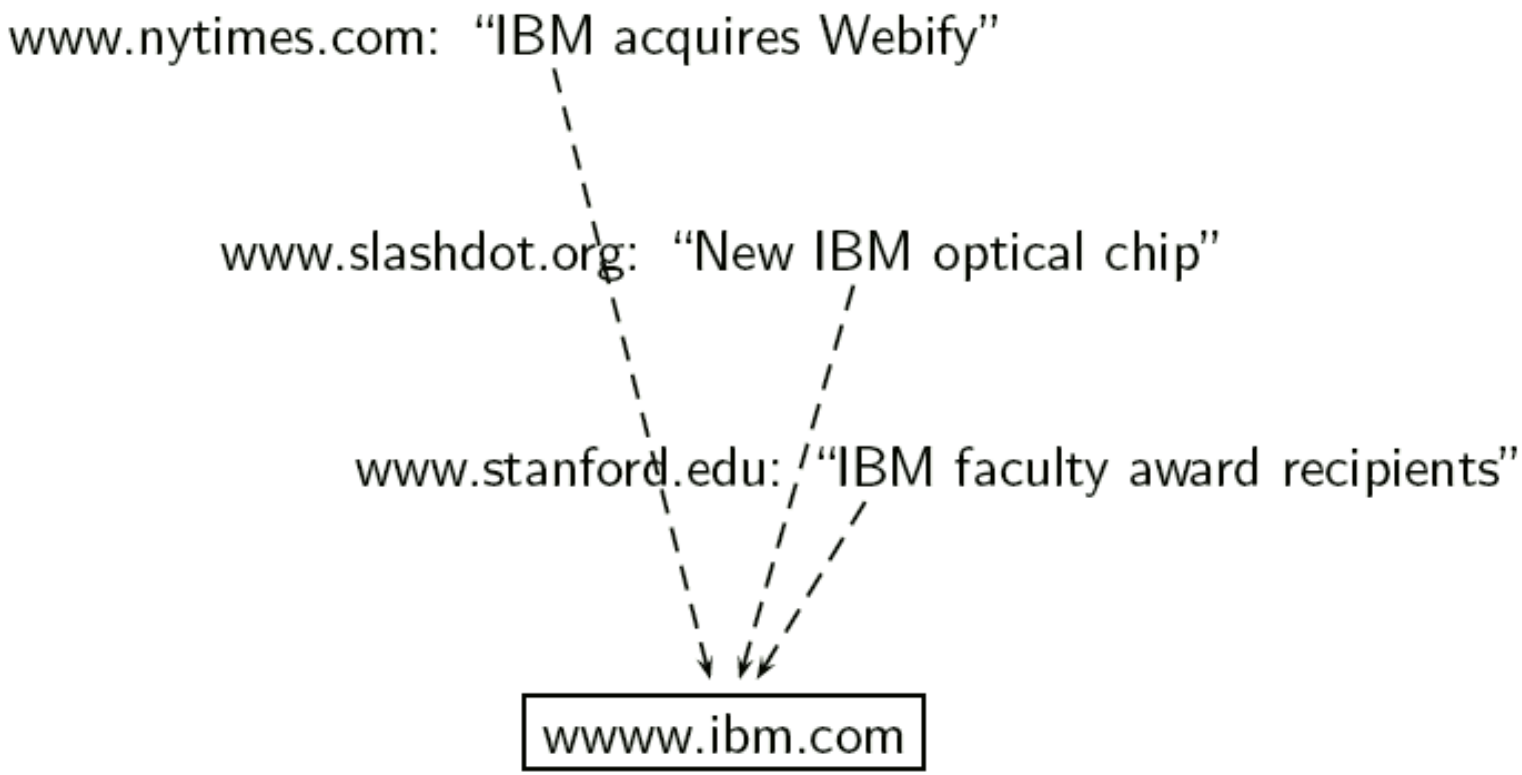


- **Assumption 1:** a hyperlink is a quality signal
  - A hyperlink between pages denotes author perceived relevance
- **Assumption 2:** The anchor text describes the target page
  - we use anchor text somewhat loosely here: the text surrounding the hyperlink. Example: “You can find cheap cars”

[document text only] vs. [document text + anchor text]

- Searching on [document text + anchor text] is often more effective than searching on [document text only].
- Example: Query ***IBM***
  - Matches IBM's copyright page
  - Matches many spam pages
  - Matches IBM wikipedia article
  - May not match IBM home page! (if IBM home page is mostly graphical)
- Searching on anchor text is better for the query IBM.
- **Represent each page by all the anchor text pointing to it.**
- In this representation, the page with the most occurrences of IBM is www.ibm.com.

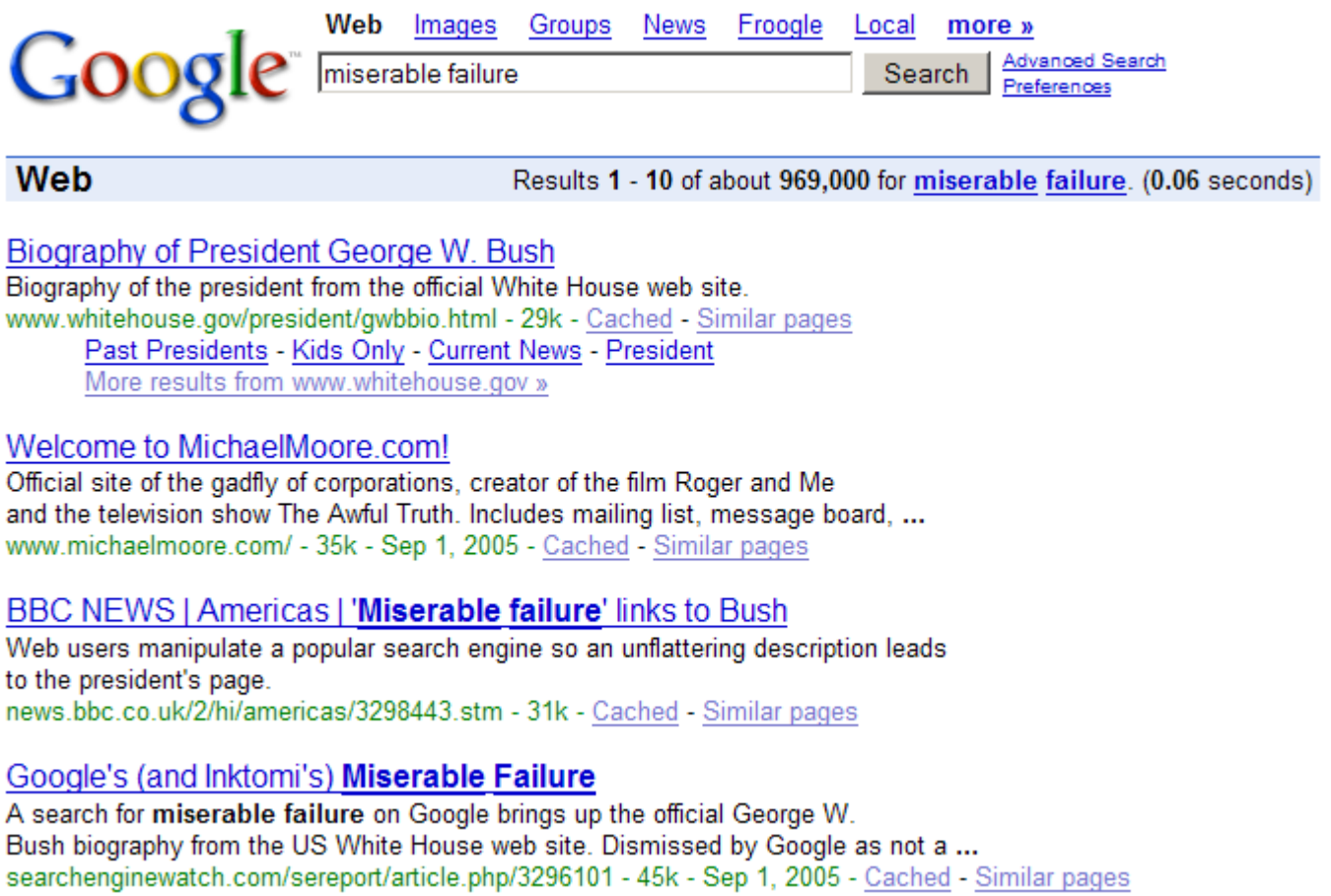
Anchor text containing *IBM* pointing to www.ibm.com



Indexing anchor text

- Thus: anchor text is often a better description of a page's content than the page itself
- Anchor text can be weighted more highly than document text (based on Assumptions 1 & 2)
- Indexing anchor text can have unexpected side effects - Google bombs.
- A Google bomb is a search with "bad" results due to maliciously manipulated anchor text
- Google introduced a new weighting function in January 2007 that fixed many Google bombs

Google bomb example



News Front Page

Africa
Americas
Asia-Pacific
Europe
Middle East
South Asia
UK
Business
Health
Science & Environment
Technology
Entertainment
Also in the news

Video and Audio

Programmes
Have Your Say
In Pictures
Country Profiles
Special Reports

RELATED BBC SITES
SPORT
WEATHER
ON THIS DAY

Last Updated: Sunday, 7 December, 2003, 15:04 GMT

E-mail this to a friend Printable version

'Miserable failure' links to Bush

George W Bush has been Google bombed.

Bush has been the target of similar pranks before

Web users entering the words "miserable failure" into the popular search engine are directed to the biography of the president on the White House website.

The trick is possible because Google searches more than just the contents of web pages - it also counts how often a site is linked to, and with what words.

Thus, members of an online community can affect the results of Google searches - called "Google bombing" - by linking their sites to a chosen one.

Weblogger Adam Mathes is credited with inventing the practice in 2001, when he used it to link the phrase "talentless hack" to a friend's website.

The search engine can be manipulated by a fairly small group of users, one report suggested.

Newsday newspaper says as few as 32 web pages with the words "miserable failure" link to the Bush biography.

The Bush administration has

> **"If you are George Bush and typed the country's name in the address bar, make sure that it is spelled correctly (IRAQ)"**
>
> Prank website

SEE ALSO:
- WMD spoof is internet hit
  04 Jul 03 | West Midlands
- Google hit by link bombers
  13 Mar 02 | Science/Nature

RELATED INTERNET LINKS:
- White House
- Google bombing

The BBC is not responsible for the content of external internet sites

TOP AMERICAS STORIES
- US lifts lid on WikiLeaks probe
- Iran scientist heads home
- Argentina legalises gay marriage

News feeds

Web Search: Pre-History

Brief (non-technical) history of Web Search

- Early keyword-based engines ca. 1995-1997
  - Altavista, Excite, Infoseek, Inktomi, Lycos,
- Paid placement ranking: Goto.com (morphed into Overture.com → Yahoo!)
  - Your search ranking depended on how much you paid
  - Auction for keywords: ***casino*** was expensive!

ALTAVISTA
Technology, Inc.

View Multimedia From Our Vantage Point

Click here for advertising information - reach millions every month!

Search the Web and Display the Results in Standard Form

Submit

Search with Digital's Alta Vista [Advanced Search]

Add your URL to the most popular Search Engines

Free Software
Download Now...

Contests
Make Me Laugh...

Creative Web
Create a Site...

FREE WEB SITES ! Create Your Personal Web Page For Free With Howdy! FREE WEB SITES !

ALTAVISTA
Technology, Inc.

[Creative][Search][Humor]

Search for information about:

in the World Wide Web Search Now

Infoseek Guide is best viewed with:

Want personalized news? Get Personal now!

Basic Search Tips:

- Click in the box above and type a few words that describe what you want to find. For example, typing **growing orchids indoors** will find sites about caring for orchids.
- If you are looking for a person or place, type the name, starting with capital letters. For example, typing **Florence Italy** will find sites about this famous city.
- These detailed search tips describe how to use the features of Infoseek Guide to find what you are looking for.
- For the broadest results, you can search the entire **World Wide Web**.
- To restrict your search to hand-picked and categorized sites, choose **Infoseek Select Sites**.
- Or just search for a category within Infoseek Select by choosing **Categories of Sites**.
- To search through Internet discussion forums (similar to bulletin boards), choose **Usenet Newsgroups**.
- To search for someone's e-mail address, choose **E-mail Addresses**.
- To search through news stories within the past month, choose **Reuters News**.
- To search through answers to Frequently Asked Questions, choose **Web FAQs**.

Explore these popular Infoseek Select topics:

- Arts & Entertainment
- Business & Finance
- Computers & Internet
- Education
- Government & Politics
- Health & Medicine
- Living
- News
- Reference
- Science & Technology
- Sports
- Travel

Try Infoseek Personal, your personalized news service

Click here to try Microsoft Money 97 FREE

It's amazing where
Go Get It will get you.

Find: Go Get It

Enhance your search.

Add Your Site to Lycos

Questions & Comments

search reviews city.net live! NEW tours

people finder maps yellow pages news

"Turbo Search!"
Download Excite Direct

Take an ExciteSeeing Tour

Excite on TV

Make your website searchable, FREE!

Excite Search: twice the power of the competition.

What: search

Where: World Wide Web

[Help]
[Advanced Search]

Excite Reviews: site reviews by the web's best editorial team.

- Arts
- Business
- Computing
- Education
- Entertainment
- Health
- Hobbies
- Life & Style
- Money
- News & Reference
- Personal Pages
- Politics & Law
- Regional
- Science
- Shopping
- Sports

Excite City.Net

Plan your weekend, your travels.

Find-A-Destination

Take me there!

Maps · Top Cities · Concierge

ExciteSeeing Tours

Choose from hundreds.

- X-Files: The truth is out there!
- Dr. Ruth's guide to safer sex
- Windows 95 shareware and freeware
- Celebrating Thanksgiving
- Investing in high-tech stocks
- New to the Net?

Excite Live!

Your news, your way.

- Latest news
- Sports scores
- Local weather
- Movie reviews
- Stock quotes
- TV listings
- Horoscopes
- Site reviews

Excite Reference

Just the facts, ma'am.

- Yellow Pages
- People Finder
- Email Lookup
- Maps
- Shareware
- Dictionary

Search the web using Google!

10 results | Google Search | I'm feeling lucky

Index contains ~25 million pages (soon to be much bigger)

About Google!

Stanford Search Linux Search

Get Google! updates monthly!

your e-mail | Subscribe | Archive

Search the web using Google!

Google Search I'm feeling lucky

Special Searches		Get Google! updates monthly:
Stanford Search	Help!	your e-mail
Linux Search	About Google!	Subscribe Archive
	Company Info	
	Google! Logos	

Jobs@Google About Google

Search the web using Google

Google Search I'm feeling lucky

Google Launches! Read the press release.

Brief (non-technical) history

- 1998+: Link-based ranking pioneered by Google
  - Blew away all early engines
  - Great user experience in search of a business model
  - Meanwhile Goto/Overture's annual revenues were nearing $1 billion
- Result: Google added paid-placement "ads" to the side, independent of search results
  - Yahoo follows suit, acquiring Overture (for paid placement) and Inktomi (for search)
- 2005+: Google gains search share, dominating in Europe and very strong in North America
  - 2009: Yahoo! and Microsoft propose combined paid search offering

Google iphone

All News Shopping Images Apps More Search tools

About 1,460,000,000 results (0.33 seconds)

Trade up to a new iPhone
Ad www.apple.com/
Trade in your current smartphone and get up to $350 in credit.
Get instant credit · Get a gift card

iPhone 6s
The only thing that's changed is everything. Learn more.

Buy now
Order now and get free shipping. Or choose free in-store pickup.

In the news

Used iPhone 6 could be the bargain you're looking for
CNET - 15 hours ago
This is especially true of Apple's **iPhone**. But when is the best time to get the great deal?

iPhone 7 Plus to Boast Dual Rear Cameras: Report
PetaPixel - 10 hours ago

Apple Loop: New iPhone Leaks, iPad Air 3's Launch Date, iOS 9.2.1 Reveals Secret iPhone Powers
Forbes - 1 day ago

More news for iphone

iPhone - Apple
www.apple.com/**iphone**/ Apple
iPhone 6s. With the most powerful technology and most intuitive operating system ever. It's here, and yours to explore.
Compare iPhone Models - Where to buy iPhone - Accessories - iPhone in Business

Apple iPhone 6s - 64 GB - Rose Gold - Verizon - CDMA/GSM
4.8 ★★★★★ 4,312 user reviews

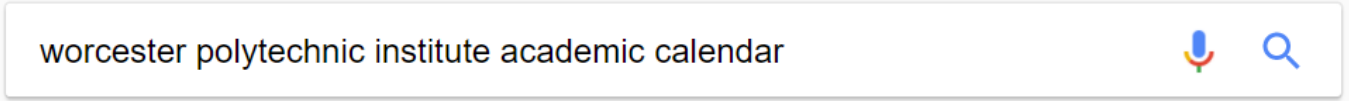
Shop now Sponsored

Rose Gold 64 GB Verizon - CDMA/GSM

$299.00 · Apple Store Free shipping
With contract

$299.99 · Best Buy Free shipping

View all sellers and prices



About 119,000 results (0.60 seconds)

Academic Calendar & Catalogs - Worcester Polytechnic Institute

https://www.wpi.edu/academics/calendar-catalogs

The information on this page is accurate as of the date of publication. However, all future **academic** calendars are reviewed annually, published for planning purposes, and are subject to change. Important Dates. Feb15. **Academic** Advising Day. 8:00 am to 11:00 pm. Feb23. Reading Day. 8:00 am to 11:00 pm. Mar2.

[PDF] Undergraduate (PDF)

https://www.wpi.edu/sites/default/files/UG_17-18_20170612.pdf

Jun 12, 2017 - **CALENDAR**. 2017-2018. S M T W R F S S M T W R F S. JUL 16 17 18 19 20 21 22. 4. 5 6 7 8 9 10. 23 24 25 26 27 28 29 FEB 11 12 13 14 15 16 17 FEBRUARY 15. ACAD. ADV. DAY. (PROJ. OPPORTUNITIES). 30 31 1 2 3 4 5. 18 19 20 21 22 23 24 FEBRUARY 23. READING/MAKEUP DAY. 6 7 8 9 10 11 ...

University Calendar - Worcester Polytechnic Institute

https://www.wpi.edu/news/calendar

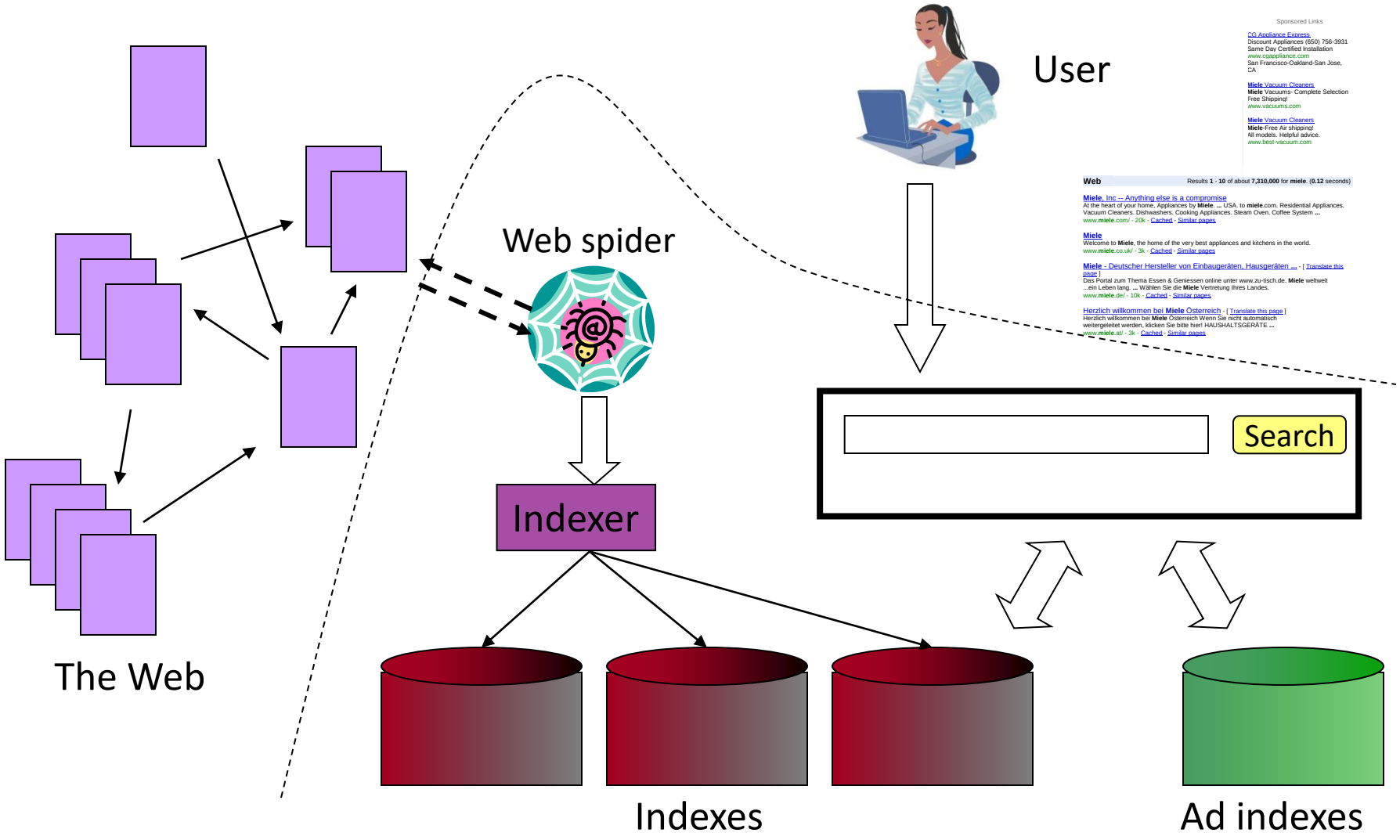
Academic Calendars · Varsity Athletics **Calendar** · Annual Events · Campus Dining · Residence Halls · Add Your Event. WPI in the World. Global Impact Program. Global Projects Program · About WPI · Bookstore · Canvas · Careers · Directories · Library · Offices · Worcester. **WORCESTER POLYTECHNIC INSTITUTE**

[PDF] undergraduate calendar 2018-2019

https://www.wpi.edu/.../Academic.../Academic-Calendars/Future%20Calendars%20-%...

UNDERGRADUATE. **CALENDAR**. 2017-2018. S M T W R F S. S M T W R F S. JUL 16 17 18 19 20 21 22. 4. 5 6 7 8 9 10. 23 24 25 26 27 28 29. FEB 11 12 13 14 15 16 17. FEBRUARY 15. ACAD. ADV. DAY. (PROJ. OPPORTUNITIES). 30 31 1 2 3 4 5. 18 19 20 21 22 23 24. FEBRUARY 23. READING DAY. 6 7 8 9 10 11 12.

Web search basics



PageRank

Link-based ranking

- Query processing with link-based ranking:
  - First retrieve all pages meeting the query (say **venture capital**)
  - Order these by their link popularity (= citation frequency, first generation)
  - . . . or by Pagerank (second generation)

- Simple link popularity (= number of inlinks of a page) is easy to spam.
- Why?

amazonmechanical turk beta
Artificial Artificial Intelligence

Your Account | HITs | Qualifications

Introduction | **Dashboard** | **Status** | **Account Settings**

Mechanical Turk is a marketplace for work.

We give businesses and developers access to an on-demand, scalable workforce.
Workers select from thousands of tasks and work whenever it's convenient.

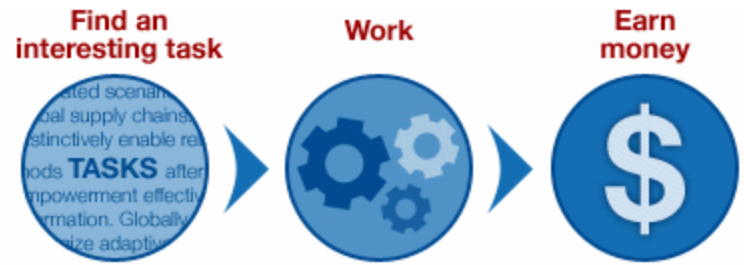
162,119 HITs available. View them now.

Make Money
by working on HITs

HITs - *Human Intelligence Tasks* - are individual tasks that you work on. Find HITs now.

As a Mechanical Turk Worker you:

- Can work from home
- Choose your own work hours
- Get paid for doing good work



Find HITs Now

or learn more about being a **Worker**

Get Results
from Mechanical Turk Workers

Ask workers to complete HITs - *Human Intelligence Tasks* - and get results using Mechanical Turk. Register Now

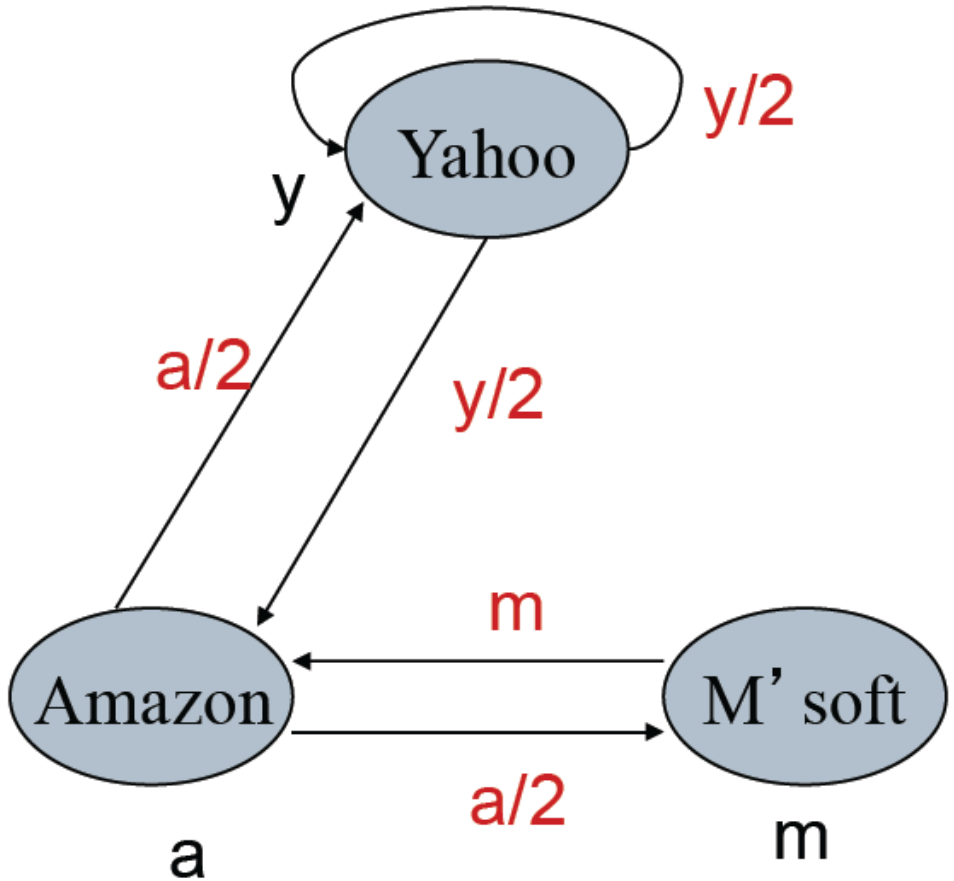
As a Mechanical Turk Requester you:

- Have access to a global, on-demand, 24 x 7 workforce
- Get thousands of HITs completed in minutes
- Pay only when you're satisfied with the results

Get Started

PageRank: Recursive formulation

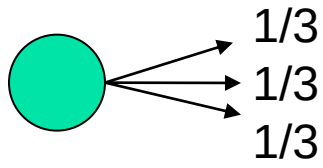
- Each link's vote is proportional to the **importance of its source page**
- If page P with importance x has n outlines, each link gets x/n votes
- Page P's own importance is the sum of the vote on its inlinks



$$y = y/2 + a/2$$
$$a = y/2 + m$$
$$m = a/2$$

PageRank basics

- Imagine a web surfer doing a random walk on the web
  - Start at a random page
  - At each step, go out of the current page along one of the links on that page, equiprobably

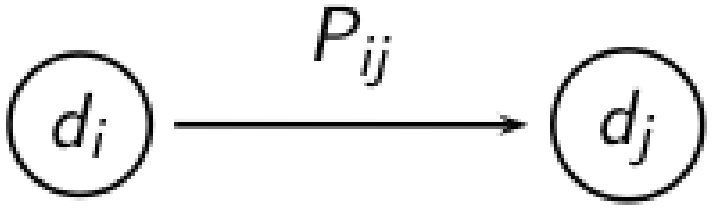


- "In the steady state" each page has a long-term visit rate - use this as the page's score.
- **PageRank = steady state probability**

  = long-term visit rate

Markov chains

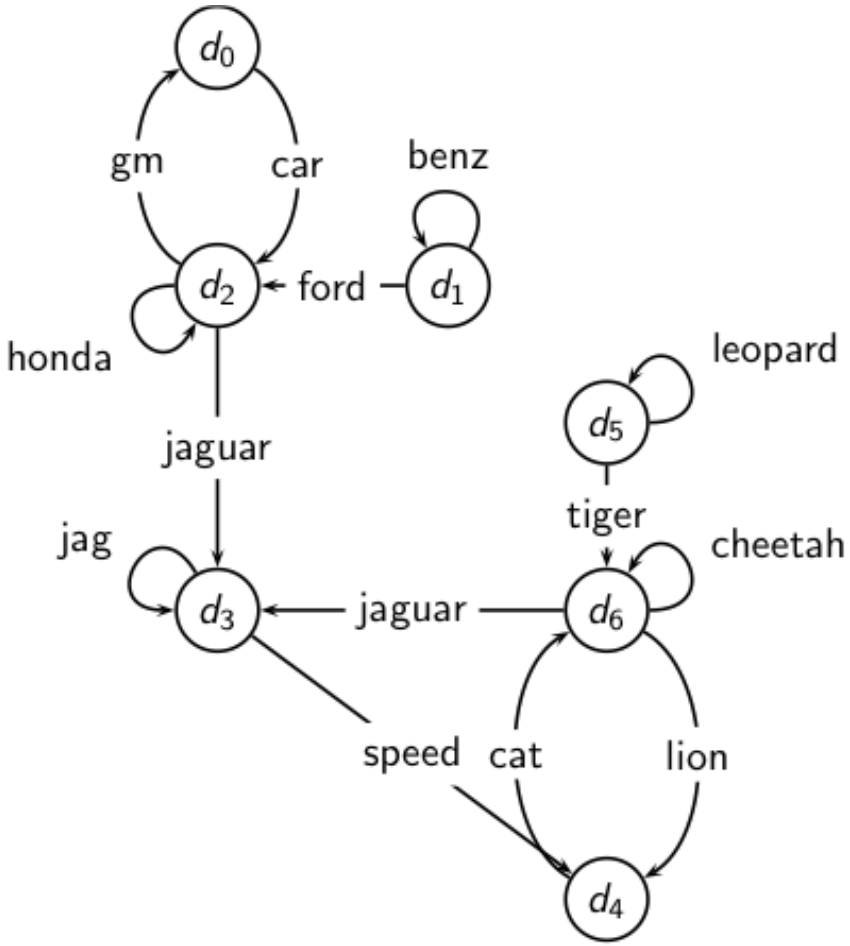
- A Markov chain consists of n states, plus an n×n transition probability matrix **P**.
- **state = page**
- At each step, we are on exactly one of the states.
- For $1 \leq i, j \leq n$, the matrix entry P_{ij} tells us the probability of j being the next state (page), given we are currently on page (state) i.



Markov chains

- Clearly, for all i, $\sum_{j=1}^{N} P_{ij} = 1$
- Markov chains are abstractions of random walks.

Example web graph



And the corresponding link matrix

	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0	0	1	0	0	0	0
d_1	0	1	1	0	0	0	0
d_2	1	0	1	1	0	0	0
d_3	0	0	0	1	1	0	0
d_4	0	0	0	0	0	0	1
d_5	0	0	0	0	0	1	1
d_6	0	0	0	1	1	0	1

Transition probability matrix P

	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0	0	1	0	0	0	0
d_1	0	1	1	0	0	0	0
d_2	1	0	1	1	0	0	0
d_3	0	0	0	1	1	0	0
d_4	0	0	0	0	0	0	1
d_5	0	0	0	0	0	1	1
d_6	0	0	0	1	1	0	1

Transition probability matrix

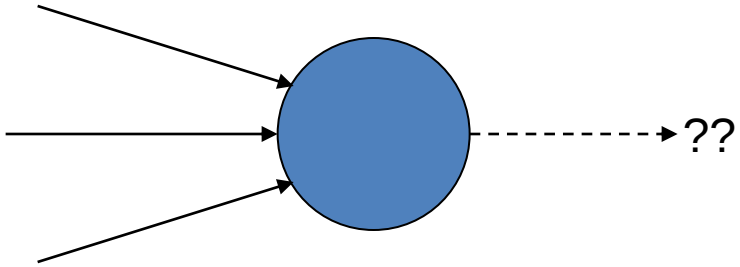
	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0.00	0.00	1.00	0.00	0.00	0.00	0.00
d_1	0.00	0.50	0.50	0.00	0.00	0.00	0.00
d_2	0.33	0.00	0.33	0.33	0.00	0.00	0.00
d_3	0.00	0.00	0.00	0.50	0.50	0.00	0.00
d_4	0.00	0.00	0.00	0.00	0.00	0.00	1.00
d_5	0.00	0.00	0.00	0.00	0.00	0.50	0.50
d_6	0.00	0.00	0.00	0.33	0.33	0.00	0.33

Long-term visit rate

- Recall: PageRank = long-term visit rate
- Long-term visit rate of page ***d*** is the probability that a web surfer is at page ***d*** at a given point in time.
- Next: what properties must hold of the web graph for the long-term visit rate to be well defined?

Not quite enough

- The web is full of dead-ends.
  - Random walk can get stuck in dead-ends.
  - Makes no sense to talk about long-term visit rates.



Teleporting

- At a dead end, jump to a random web page.
- At any non-dead end, with probability 10%, jump to a random web page.
  - With remaining probability (90%), go out on a random link.
  - 10% - a parameter.

Teleporting Matrix

- Recall: At a dead end, jump to a random web page

	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	1/7	1/7	1/7	1/7	1/7	1/7	1/7
d_1	1/7	1/7	1/7	1/7	1/7	1/7	1/7
d_2	1/7	1/7	1/7	1/7	1/7	1/7	1/7
d_3	1/7	1/7	1/7	1/7	1/7	1/7	1/7
d_4	1/7	1/7	1/7	1/7	1/7	1/7	1/7
d_5	1/7	1/7	1/7	1/7	1/7	1/7	1/7
d_6	1/7	1/7	1/7	1/7	1/7	1/7	1/7

Result of teleporting

- With teleporting, we cannot get stuck in a dead end
- There is a long-term rate at which any page is visited (not obvious, will show this).
- How do we compute this visit rate?

Formalization of "visit": Probability vectors

- A probability (row) vector **x** = $(x_1, \ldots x_n)$ tells us where the walk is at any point.
- E.g., (000…1…000) means we're in state *i*.
  1 *i* *n*

- More generally, the vector **x** = $(x_1, \ldots x_n)$ means the walk is in state *i* with probability x_i.

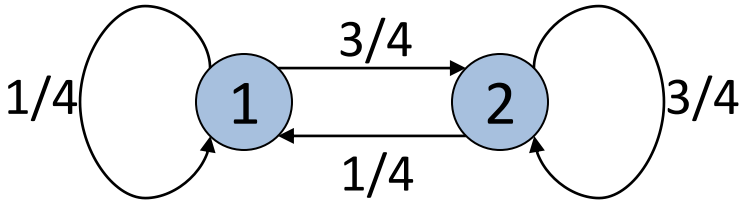
$$\sum_{i=1}^{n} x_i = 1.$$

Change in probability vector

- If the probability vector is $\mathbf{x} = (x_1, \ldots x_n)$ at this step, what is it at the next step?
- Recall that row *i* of the transition prob. Matrix **P** tells us where we go next from state *i*.
- So from **x**, our next state is distributed as **xP**.

Steady state example

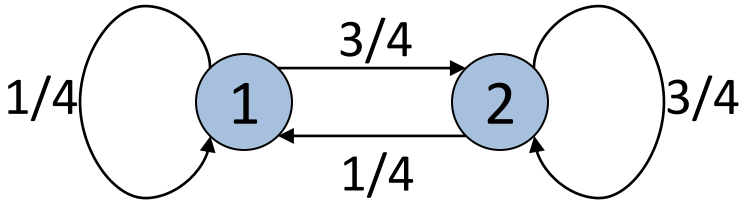
- The steady state looks like a vector of probabilities **a** = $(a_1, \ldots a_n)$:
- a_i is the probability that we are in state i.



What is the steady state in this example?

Steady state example

- The steady state looks like a vector of probabilities **a** = $(a_1, \ldots a_n)$:
- a_i is the probability that we are in state i.



For this example, $a_1=1/4$ and $a_2=3/4$.

How to compute the steady-state?

- Recall, regardless of where we start, we eventually reach the steady state **a**.
- Start with any distribution (say **x**=(*10…0*)).
- After one step, we're at **xP**;
- after two steps at $\mathbf{xP}^2$, then $\mathbf{xP}^3$ and so on.
- "Eventually" means for "large" k, $\mathbf{xP}^k = \mathbf{a}$.
- Algorithm: multiply **x** by increasing powers of **P** until the product looks stable.
- This is called the power method

Power method: example

Two-node example: $\vec{x} = (0.5, 0.5)$, $P = \begin{pmatrix} 0.25 & 0.75 \\ 0.25 & 0.75 \end{pmatrix}$

$\vec{x}P = (0.25, 0.75) = \vec{x}_2$

$\vec{x}_2 P = (0.25, 0.75)$

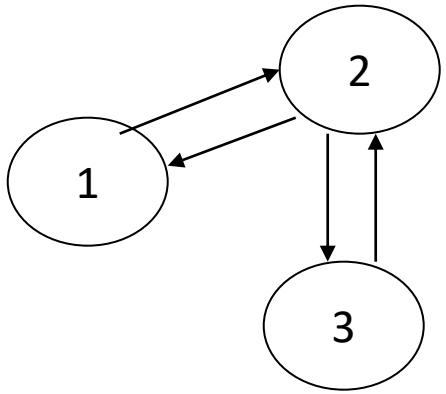
Convergence in one iteration!

Exercise on PageRank

Transition probability matrix of a surfer's walk with teleportation:

P = (1- α) * transition matrix + α * teleporting matrix

- Consider a Web graph with three nodes 1, 2, and 3. The links are as follows: 1->2, 3->2, 2->1, 2->3. Write down the transition probability matrices P and pagerank scores for the surfer's walk with teleporting, with the value of teleport probability α=0.5.



P =

0	1	0
1	0	1
0	1	0

Each 1 divied by the number of ones in this row

(1-α)*

0	1	0
½	0	½
0	1	0

+

α*

1/3	1/3	1/3
1/3	1/3	1/3
1/3	1/3	1/3

=

1/6	2/3	1/6
5/12	1/6	5/12
1/6	2/3	1/6

Exercise on PageRank (Cont'd)

Remember

$\vec{x}_1 = \vec{x}_0 P$

$\vec{x}_2 = \vec{x}_1 P$

$\vec{x}_3 = \vec{x}_2 P$

…

…

…

Until converged

$\vec{x}_0 =$

1	0	0

P=

1/6	2/3	1/6
5/12	1/6	5/12
1/6	2/3	1/6

$\vec{x}_1 =$

1/6	2/3	1/6

$\vec{x}_2 =$

1/3	1/3	1/3

$\vec{x}_3 =$

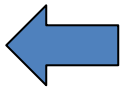
1/4	1/2	1/4

…

…

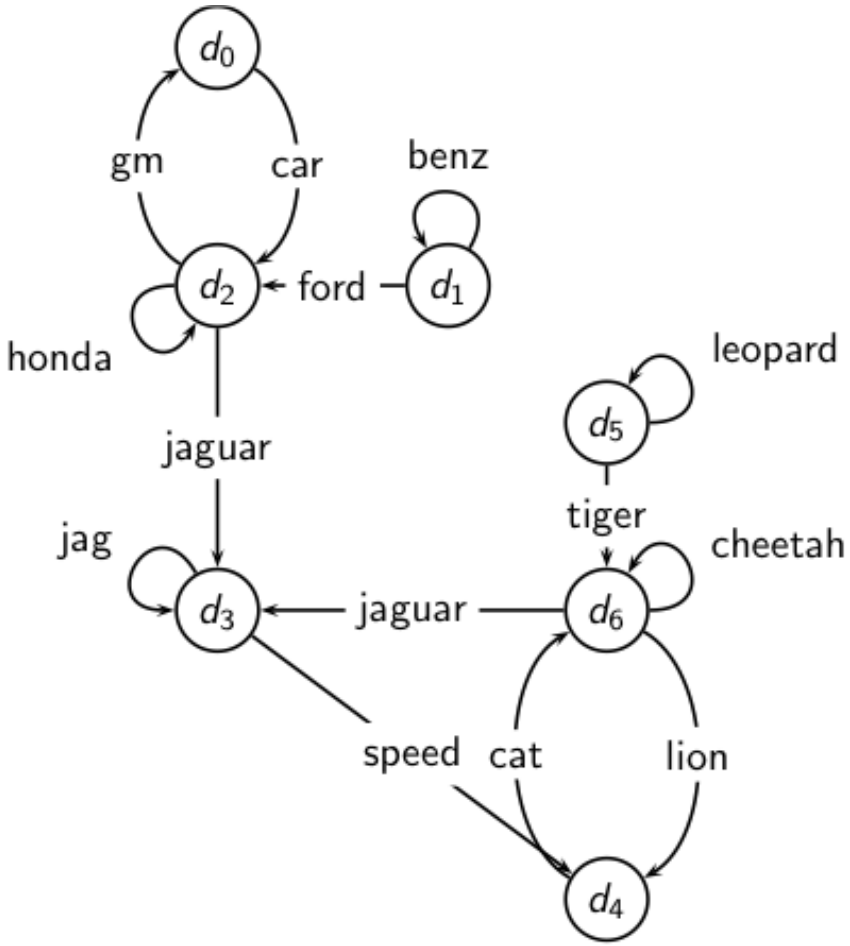
$\vec{x}_k =$

5/18	4/9	5/18



converged

Example web graph



And the corresponding link matrix

	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0	0	1	0	0	0	0
d_1	0	1	1	0	0	0	0
d_2	1	0	1	1	0	0	0
d_3	0	0	0	1	1	0	0
d_4	0	0	0	0	0	0	1
d_5	0	0	0	0	0	1	1
d_6	0	0	0	1	1	0	1

Transition matrix with teleporting

	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0.00	0.00	1.00	0.00	0.00	0.00	0.00
d_1	0.00	0.50	0.50	0.00	0.00	0.00	0.00
d_2	0.33	0.00	0.33	0.33	0.00	0.00	0.00
d_3	0.00	0.00	0.00	0.50	0.50	0.00	0.00
d_4	0.00	0.00	0.00	0.00	0.00	0.00	1.00
d_5	0.00	0.00	0.00	0.00	0.00	0.50	0.50
d_6	0.00	0.00	0.00	0.33	0.33	0.00	0.33

$\alpha = 0.14$

P =

	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0.02	0.02	0.88	0.02	0.02	0.02	0.02
d_1	0.02	0.45	0.45	0.02	0.02	0.02	0.02
d_2	0.31	0.02	0.31	0.31	0.02	0.02	0.02
d_3	0.02	0.02	0.02	0.45	0.45	0.02	0.02
d_4	0.02	0.02	0.02	0.02	0.02	0.02	0.88
d_5	0.02	0.02	0.02	0.02	0.02	0.45	0.45
d_6	0.02	0.02	0.02	0.31	0.31	0.02	0.31

Power method convergence

	x	xP^1	xP^2	xP^3	xP^4	xP^5	xP^6	xP^7	xP^8	xP^9	xP^{10}	xP^{11}	xP^{12}	xP^{13}
d_0	0.14	0.06	0.09	0.07	0.07	0.06	0.06	0.06	0.06	0.05	0.05	0.05	0.05	0.05
d_1	0.14	0.08	0.06	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
d_2	0.14	0.25	0.18	0.17	0.15	0.14	0.13	0.12	0.12	0.12	0.12	0.11	0.11	0.11
d_3	0.14	0.16	0.23	0.24	0.24	0.24	0.24	0.25	0.25	0.25	0.25	0.25	0.25	0.25
d_4	0.14	0.12	0.16	0.19	0.19	0.20	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.21
d_5	0.14	0.08	0.06	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
d_6	0.14	0.25	0.23	0.25	0.27	0.28	0.29	0.29	0.30	0.30	0.30	0.30	0.31	0.31

Pagerank summary

- Preprocessing:
  - Given graph of links, build matrix **P**.
  - From it compute **a**.
  - The entry a_i is a number between 0 and 1: the pagerank of page *i*.
- Query processing:
  - Retrieve pages meeting query.
  - Rank them by their pagerank.
  - Order is **query-*independent***.

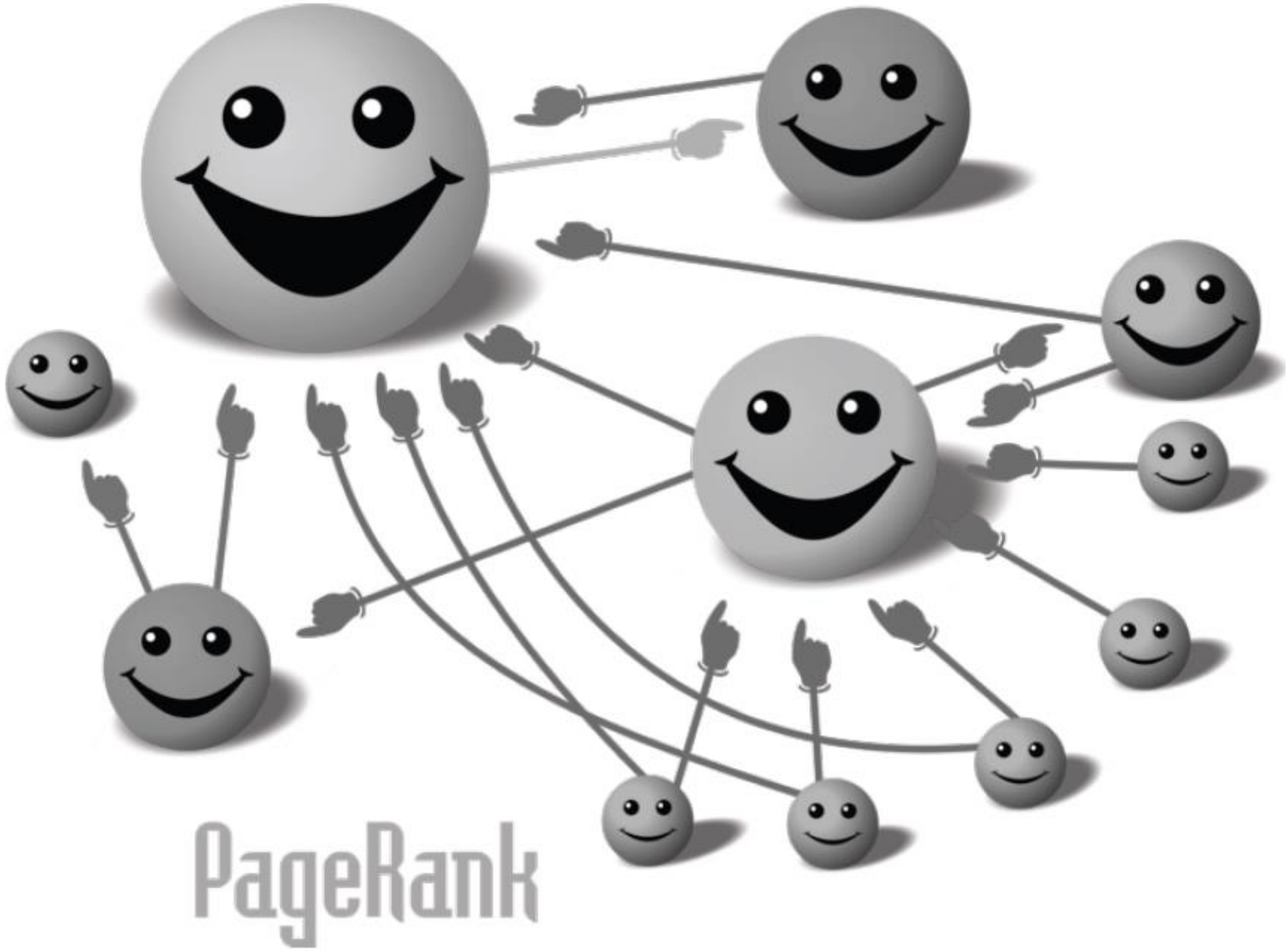
PageRank issues

- Real surfers are not random surfers – Markov model is not a good model of surfing.
  - Issues: back button, short vs. long paths, bookmarks, directories – and search!
- Simple PageRank ranking (as described on previous slide) produces bad results for many pages.
  - Consider the query ***video service***
  - The Yahoo home page (i) has a very high PageRank and (ii) contains both words.
  - If we rank all Boolean hits according to PageRank, then the Yahoo home page would be top-ranked.
  - Clearly not desirable
- In practice: rank according to weighted combination of many factors, including raw text match, anchor text match, PageRank and many other factors

How important is PageRank?

- Frequent claim: PageRank is the most important component of web ranking.
- The reality:
  - There are several components that are at least as important: e.g., anchor text, indexing , zone weighting, phrases ...
- Rumor has it that PageRank in his original form (as presented here) now has a negligible impact on ranking!
- However, variants of a page's PageRank are still an essential part of ranking.
- Addressing link spam is difficult and crucial.

What is PageRank?



HITS: Hubs & Authorities

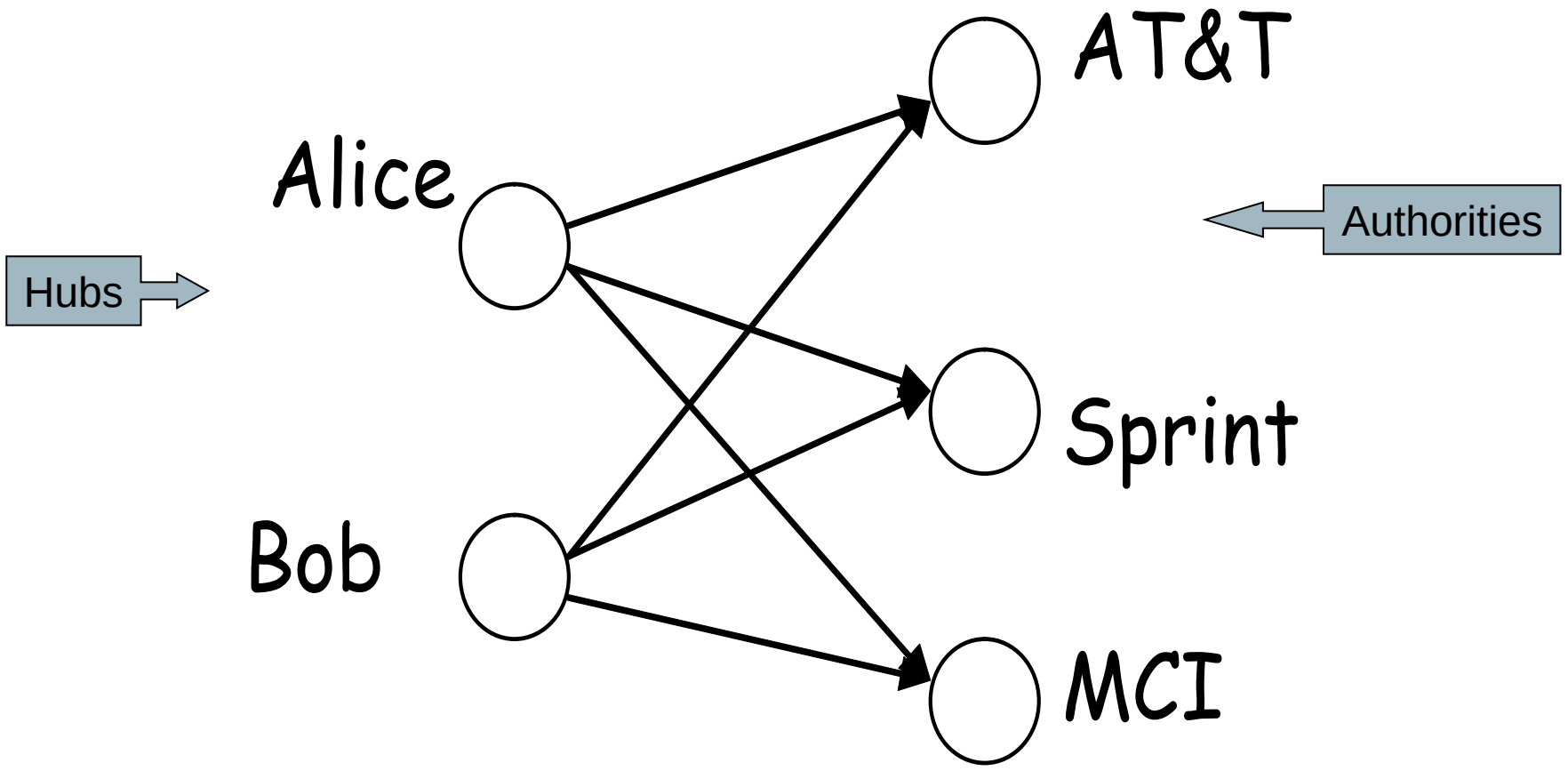
HITS – Hyperlink-Induced Topic Search

- Premise: there are two different types of relevance on the web.
- Relevance type 1: Hubs. A hub page is a good list of links to pages answering the information need.
  - E.g, for query [chicago bulls]: Bob's list of recommended resources on the Chicago Bulls sports team
- Relevance type 2: Authorities. An authority page is a direct answer to the information need.
  - The home page of the Chicago Bulls sports team
  - By definition: Links to authority pages occur repeatedly on hub pages.
- Most approaches to search (including PageRank ranking) don't make the distinction between these two very different types of relevance.

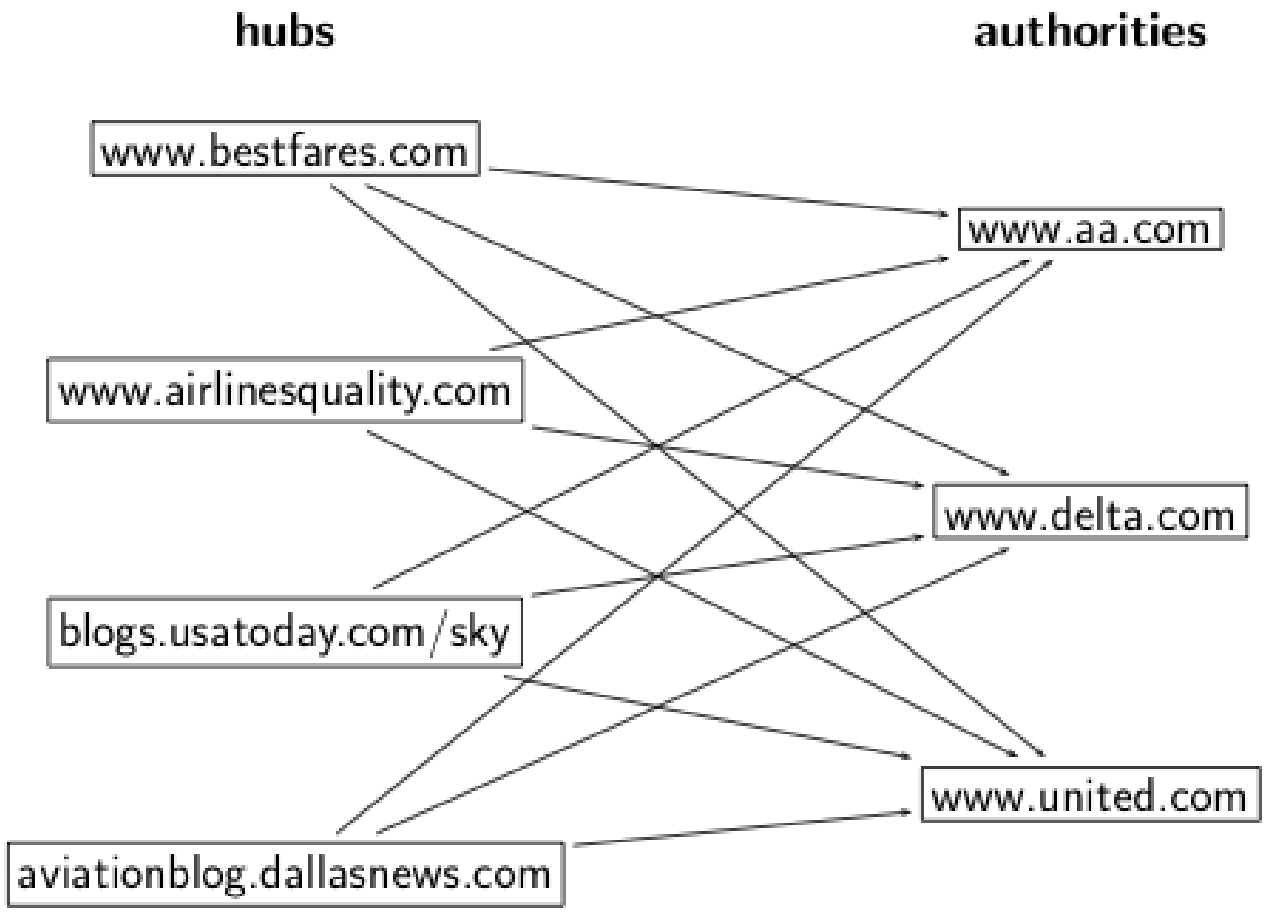
Hubs and authorities : Definition

- Thus, a good hub page for a topic *points* to many authority pages for that topic.
- A good authority page for a topic is *pointed* to by many hub pages for that topic.
- Circular definition – we will turn this into an iterative computation.

The hope



Long distance telephone companies


hubs
authorities
www.bestfares.com
www.airlinesquality.com
blogs.usatoday.com/sky
aviationblog.dallasnews.com
www.aa.com
www.delta.com
www.united.com

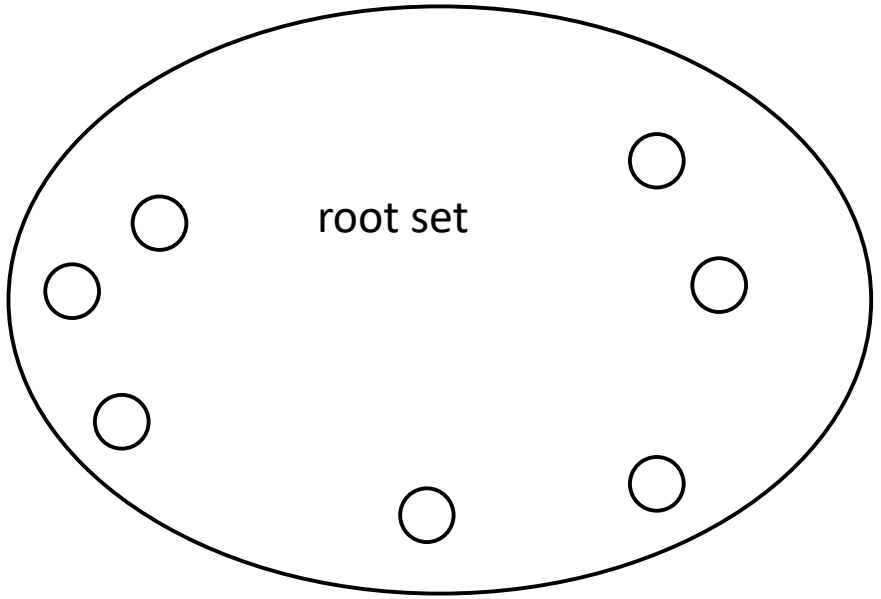
High-level scheme

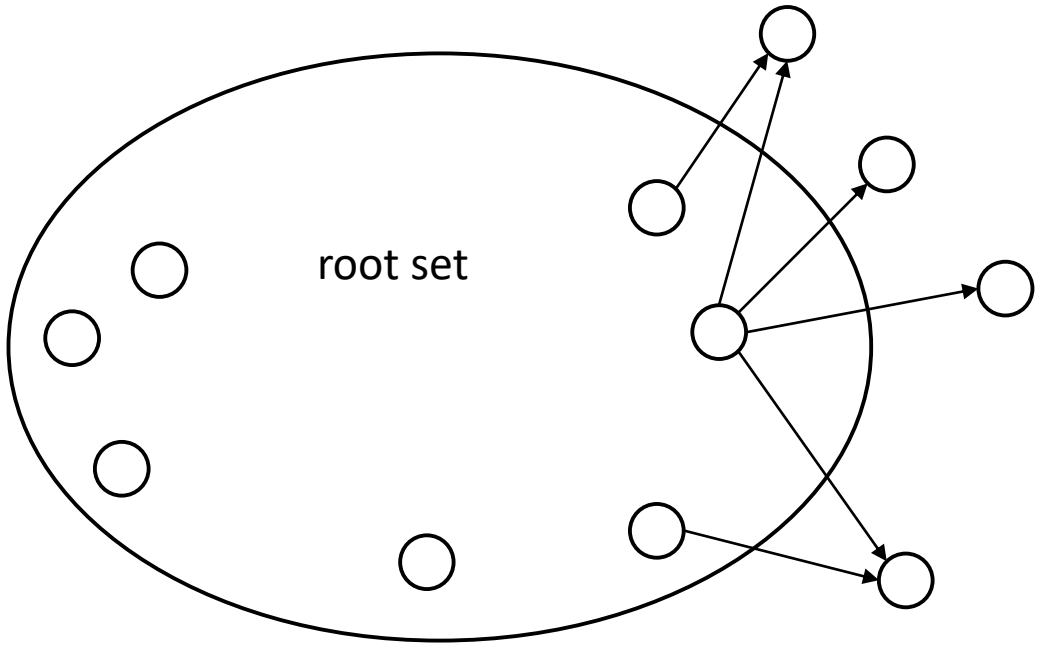
- Extract from the web a base set of pages that *could* be good hubs or authorities.
- From these, identify a small set of top hub and authority pages;

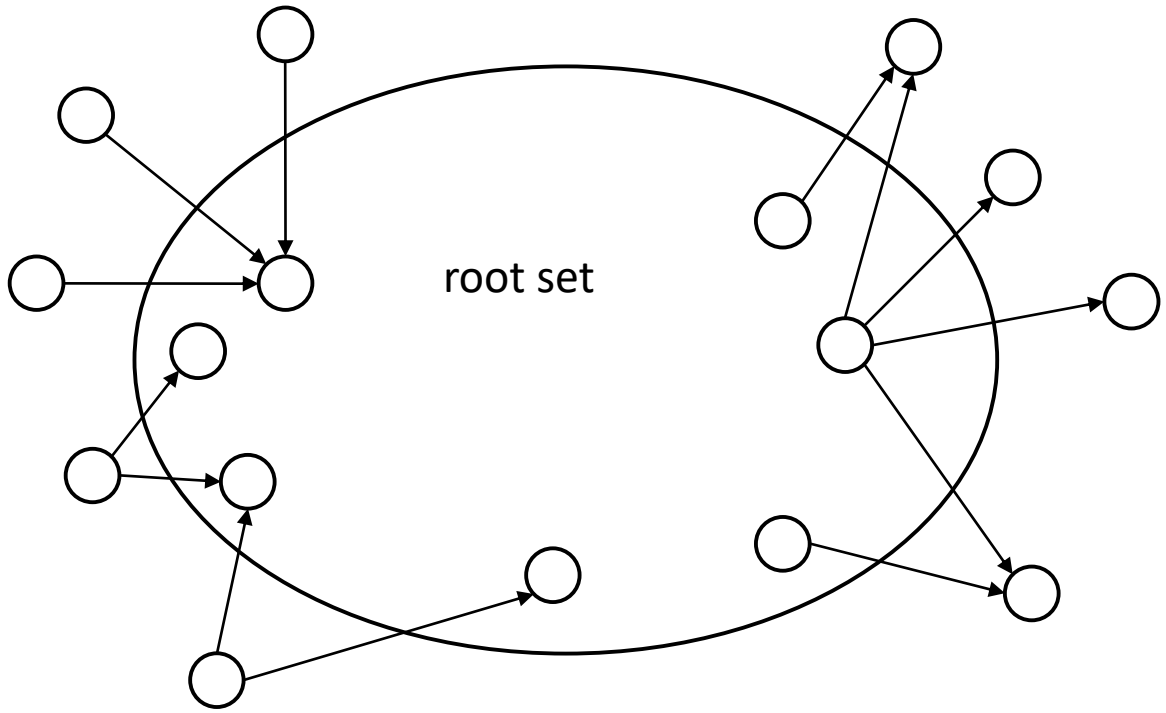
  →iterative algorithm.

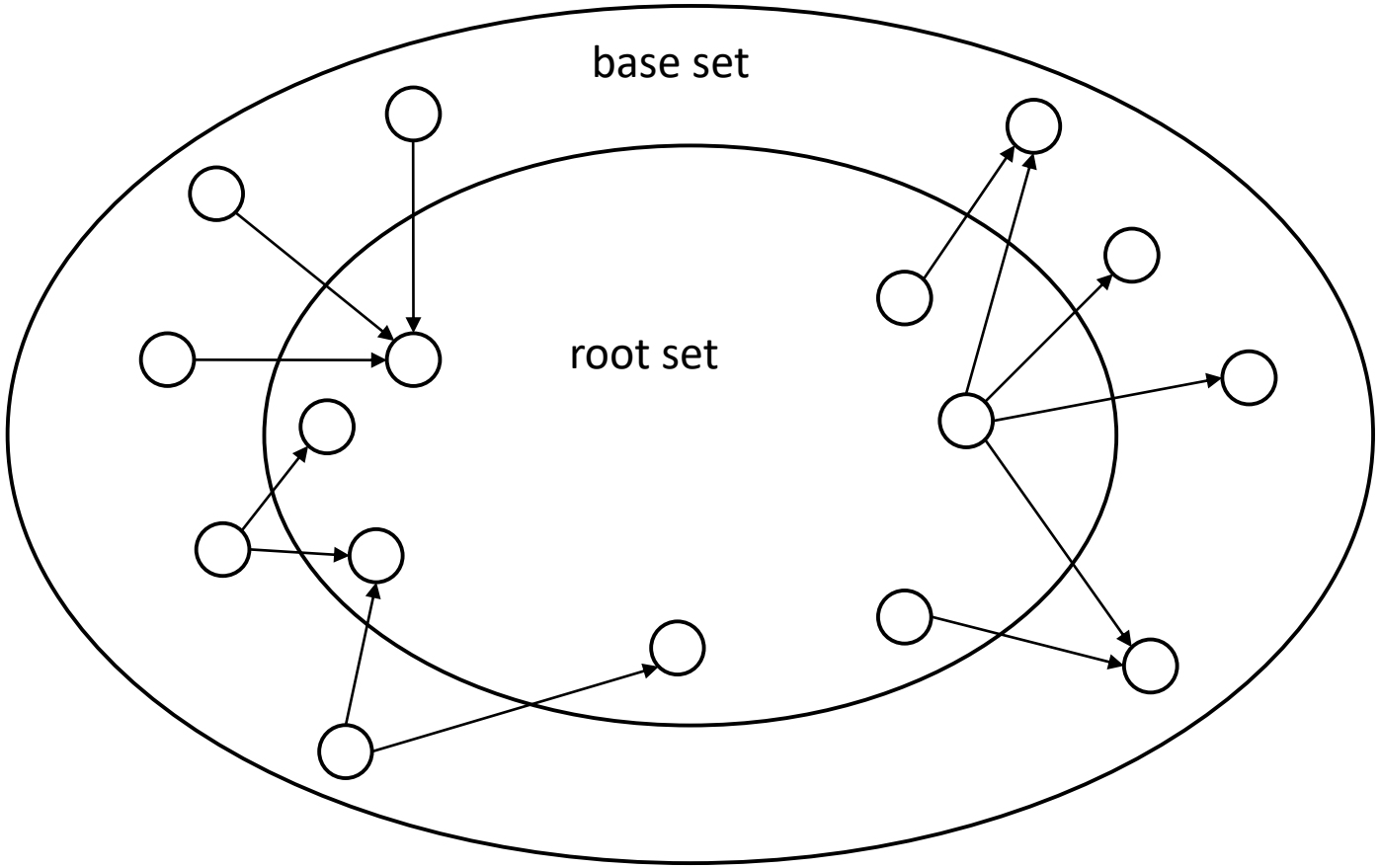
Root set and base set

- Do a regular web search first
- Call the search result the root set
- Find all pages that are linked to or link to pages in the root set
- Call first larger set the base set
- Finally, compute hubs and authorities for the base set (which we'll view as a small web graph)


root set


root set


root set


base set
root set

Root set and base set

- Root set typically has 200-1000 nodes.
- Base set may have up to 5000 nodes.
- Computation of base set, as shown on previous slide:
  - Follow outlinks by parsing the pages in the root set
  - Find *x*'s inlinks by searching for all pages containing a link to *x*
  - This assumes our inverted index supports search for links (in addition to terms)

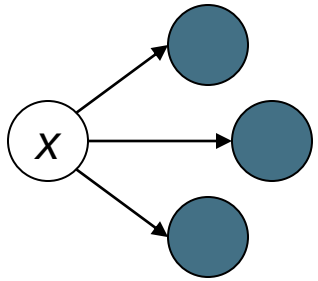
Hub and authority scores

- Compute for each page x in the base set a hub score $h(x)$ and an authority score $a(x)$
- Initialization: for all x: $h(x) \leftarrow 1$, $a(x) \leftarrow 1$;
- Iteratively update all $h(x)$, $a(x)$
- After convergence:
  - Output pages with highest $h()$ scores as top hubs
  - highest $a()$ scores as top authorities

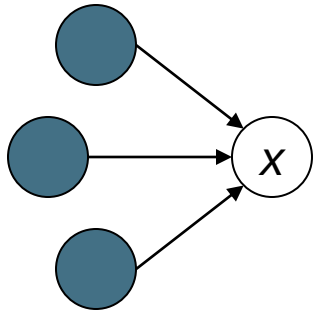
Iterative update

- Repeat the following updates, for all *x*:

$$h(x) \leftarrow \sum_{x \mapsto y} a(y)$$



$$a(x) \leftarrow \sum_{y \mapsto x} h(y)$$



Scaling

- To prevent the *a()* and *h()* values from getting too big, can scale down after each iteration.
- Scaling factor doesn't really matter:
- we only care about the ***relative*** values of the scores.

How many iterations?

- Claim: relative values of scores will converge after a few iterations:
  - in fact, suitably scaled, *h()* and *a()* scores settle into a steady state!
- We only require the relative orders of the *h()* and *a()* scores - not their absolute values.

- In practice, ~5 iterations get you close to stability.

Japan Elementary Schools

Hubs

- schools
- LINK Page-13
- “ú–{‚ÌŠwZ
- ā‰„¬ŠwZƒz[ƒƒy[ƒW
- 100 Schools Home Pages (English)
- K-12 from Japan 10/...rnet and Education)
- http://www...iglobe.ne.jp/~IKESAN
- ‚l‚f‚j¬ŠwZ‚U”N‚P ‘g•¨Œê
- ÒŠ—’ ¬— § ÒŠ—“Œ¬ŠwZ
- Koulutus ja oppilaitokset
- TOYODA HOMEPAGE
- Education
- Cay's Homepage(Japanese)
- –y“ì¬ŠwZ‚Ìƒz[ƒƒy[ƒW
- UNIVERSITY
- ‰J—³¬ŠwZ DRAGON97-TOP
- Â‰ª¬ŠwZ‚T”N‚P ‘gƒz[ƒƒy[ƒW
- ¶µ°é¼ÂÁ© ¥á¥Ë¥å¡¼ ¥á¥Ë¥å¡¼

Authorities

- The American School in Japan
- The Link Page
- ‰ª è ž—§ˆä“c¬ŠwZƒz[ƒƒy[ƒW
- Kids' Space
- ˆÀé ž—§ˆÀé¼•”¬ŠwZ
- ‹{é ³ˆç ‘åŠw••®¬ŠwZ
- KEIMEI GAKUEN Home Page (Japanese)
- Shiranuma Home Page
- fuzoku-es.fukui-u.ac.jp
- welcome to Miasa E&J school
- •_“Þì Œ § Ē‰¡•ls— § ’ †ì¼¬ŠwZ‚Ìƒy
- http://www...p/~m_maru/index.html
- fukui haruyama-es HomePage
- Torisu primary school
- goo
- Yakumo Elementary,Hokkaido,Japan
- FUZOKU Home Page
- Kamishibun Elementary School...

Things to note

- Pulled together good pages regardless of language of page content.
- Use *only* link analysis after base set assembled
  - iterative scoring is query-independent.
- Downside: Iterative computation after text index retrieval - significant overhead.

Hub/authority vectors

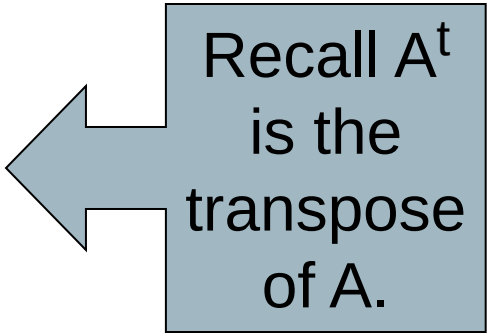
- View the hub scores *h()* and the authority scores *a()* as vectors with *n* components.
- Recall the iterative updates

$$h(x) \leftarrow \sum_{x \mapsto y} a(y)$$

$$a(x) \leftarrow \sum_{y \mapsto x} h(y)$$

Rewrite in matrix form

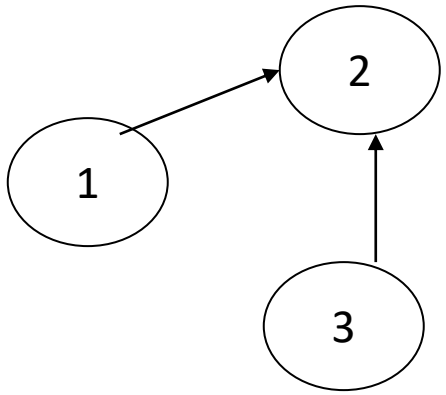
- **h**=**Aa**.
- **a**=**A**^{t}**h**.



- *A is a square* matrix with one row and one column for each page in the subset
  - *Aij is 1 if there is a hyperlink from page i to page j, and 0 otherwise*

Exercise on HITS

- Consider a Web graph with three nodes 1, 2, and 3. The links are as follows: 1->2, 3->2.



$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad A^T = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\vec{h}_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \qquad \vec{a}_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Remember

$$\vec{h}_1 = A\vec{a}_0 \qquad \vec{a}_1 = A^T\vec{h}_0$$
$$\vec{h}_2 = A\vec{a}_1 \qquad \vec{a}_2 = A^T\vec{h}_1$$
$$\vec{h}_3 = A\vec{a}_2 \qquad \vec{a}_3 = A^T\vec{h}_2$$

…

Until converged

$$\vec{h}_1 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \qquad \vec{a}_1 = \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix}$$

$$\vec{h}_2 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \qquad \vec{a}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Normalization

$$\vec{h}_1 = \begin{bmatrix} 1/2 \\ 0 \\ 1/2 \end{bmatrix} \qquad \vec{a}_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\vec{h}_2 = \begin{bmatrix} 1/2 \\ 0 \\ 1/2 \end{bmatrix} \qquad \vec{a}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

converged

PageRank vs. HITS: Discussion

- PageRank can be precomputed, HITS has to be computed at query time.
  - HITS is too expensive in most application scenarios.
- We could also apply HITS to the entire web and PageRank to a small base set.
- On the web, a good hub almost always is also a good authority.
- The actual difference between PageRank ranking and HITS ranking is therefore not as large as one might expect.

Authoritative Sources in a Hyperlinked Environment*

Jon M. Kleinberg †

Abstract

The network structure of a hyperlinked environment can be a rich source of information about the content of the environment, provided we have effective means for understanding it. We develop a set of algorithmic tools for extracting information from the link structures of such environments, and report on experiments that demonstrate their effectiveness in a variety of contexts on the World Wide Web. The central issue we address within our framework is the distillation of broad search topics, through the discovery of "authoritative" information sources on such topics. We propose and test an algorithmic formulation of the notion of authority, based on the relationship

Crowdturfers, Campaigns, and Social Media: Tracking and Revealing Crowdsourced Manipulation of Social Media

Kyumin Lee*, **Prithivi Tamilarasan***, **James Caverlee**
Texas A&M University
College Station, TX 77843
{kyumin, prithivi, caverlee}@cse.tamu.edu

Abstract

Crowdturfing has recently been identified as a sinister counterpart to the enormous positive opportunities of crowdsourcing. Crowdturfers leverage human-powered crowdsourcing platforms to spread malicious URLs in social media, form "astroturf" campaigns, and manipulate search engines, ultimately degrading the quality of online information and threatening the usefulness of these systems. In this paper we present a framework for "pulling back the curtain" on crowdturfers to reveal their underlying ecosystem. Concretely, we analyze the types of malicious tasks and the properties of requesters and

for the government or commercial products, as well as disparage rivals (Sterling 2010; Wikipedia 2013). Mass organized crowdturfers are also targeting popular services like iTunes (Chan 2012) and attracting the attention of US intelligence operations (Fielding and Cobain 2011). And increasingly, these campaigns are being launched from commercial crowdsourcing sites, potentially leading to the commoditization of large-scale turfing campaigns. In a recent study of the

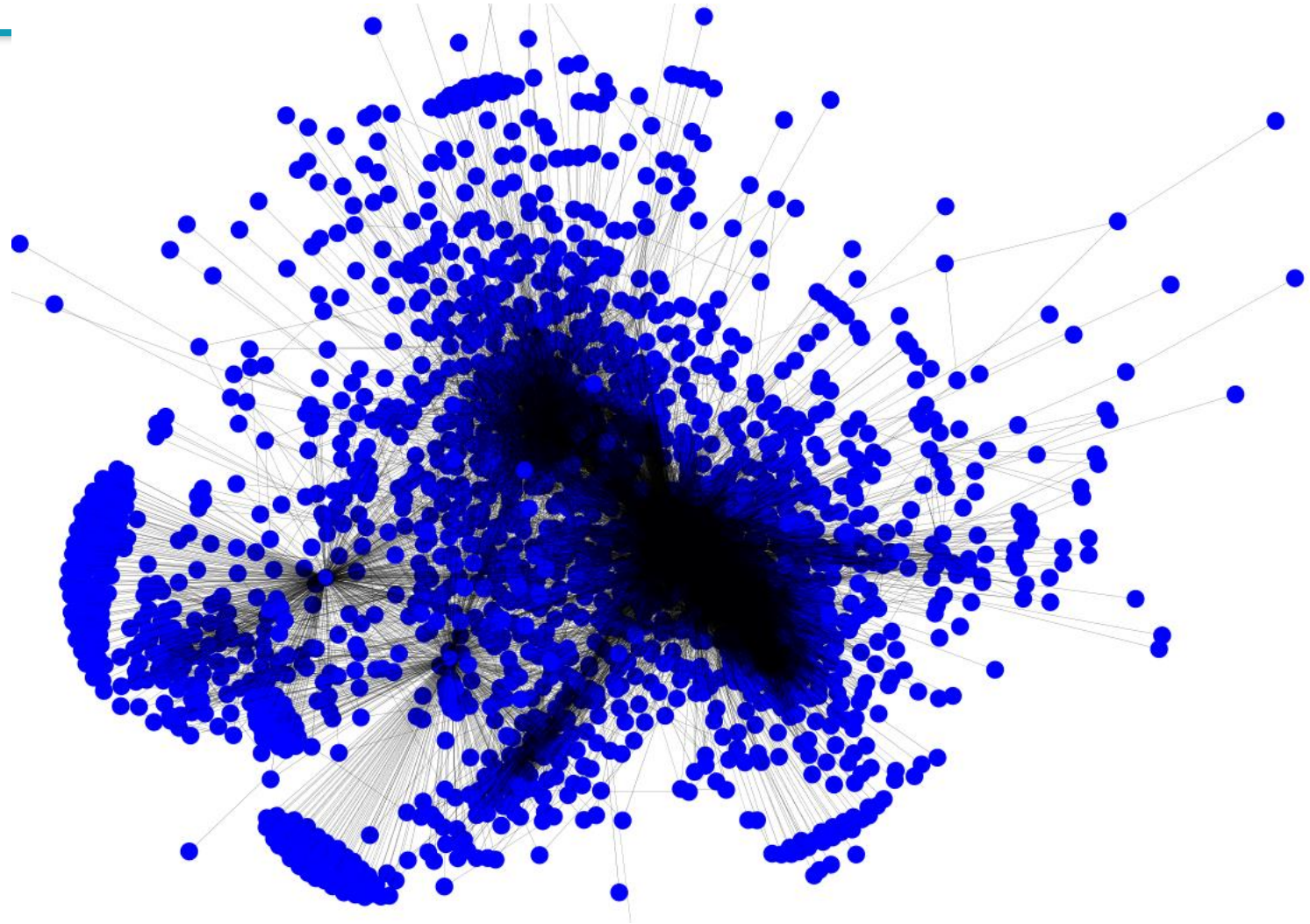
Hubs and Authorities. We next examine who in work is significant. Concretely, we adopted the wel HITS (Kleinberg 1999) algorithm to identify t (workers who follow many other workers) and au (workers who are followed by many other worker network:

$$\overrightarrow{a} \leftarrow A^T \overrightarrow{h}$$

$$\overrightarrow{h} \leftarrow A \overrightarrow{a}$$

where $\overrightarrow{h}$ and $\overrightarrow{a}$ denote the vectors of all hub and all authority scores, respectively. A is a square matrix with one row and one column for each worker (user) in the worker graph. If there is an edge between worker i and worker j, the entry A_{ij} is 1 and otherwise 0. We iterate the computation of $\overrightarrow{h}$ and $\overrightarrow{a}$ until both $\overrightarrow{h}$ and $\overrightarrow{a}$ are converged. We initialized each worker's hub and authority scores as $1/n$ – where n is the number of workers in the graph – and then computed HITS until the scores converged.

Twitter workers' following-follower relationship



Screen Name	|Followings|	|Followers|	|Tweets|
NannyDotNet	1,311	753	332
_Woman_health	210,465	207,589	33,976
Jet739	290,624	290,001	22,079
CollChris	300,385	300,656	8,867
familyfocusblog	40,254	39,810	22,094
tinastullracing	171,813	184,039	73,004
drhenslin	98,388	100,547	10,528
moneyartist	257,773	264,724	1,689
pragmaticmom	30,832	41,418	21,843
Dede_Watson	37,397	36,833	47,105

Table 6: Top-10 hubs of the workers.

Screen Name	|Followings|	|Followers|	|Tweets|
NannyDotNet	1,311	753	332
_Woman_health	210,465	207,589	33,976
CollChris	300,385	300,656	8,867
familyfocusblog	40,254	39,810	22,094
tinastullracing	171,813	184,039	73,004
pragmaticmom	30,832	41,418	21,843
Jet739	290,624	290,001	22,079
moneyartist	257,773	264,724	1,689
drhenslin	98,388	100,547	10,528
ceebee308	283,301	296,857	169,061

Table 7: Top-10 authorities of the workers.

Understanding and Combating Link Farming in the Twitter Social Network

Saptarshi Ghosh
IIT Kharagpur, India

Bimal Viswanath
MPI-SWS, Germany

Farshad Kooti
MPI-SWS, Germany

Naveen K. Sharma
IIT Kharagpur, India

Gautam Korlam
IIT Kharagpur, India

Fabricio Benevenuto
UFOP, Brazil

Niloy Ganguly
IIT Kharagpur, India

Krishna P. Gummadi
MPI-SWS, Germany

ABSTRACT

Recently, Twitter has emerged as a popular platform for discovering real-time information on the Web, such as news stories and people's reaction to them. Like the Web, Twitter has become a target for *link farming*, where users, especially spammers, try to acquire large numbers of follower links in the social network. Acquiring followers not only increases

Web, such as current events, news stories, and people's opinion about them. Traditional media, celebrities, and marketers are increasingly using Twitter to directly reach audiences in the millions. Furthermore, millions of individual users are sharing the information they discover over Twitter, making it an important source of breaking news during emergencies like revolutions and disasters [17, 23]. Recent

Keywords

Twitter, spam, link farming, Pagerank, Collusionrank

Algorithm 1 Coll

Input: network, G;
biased Pagerank, c
Output: Collusionr
initialize score vect

$d($

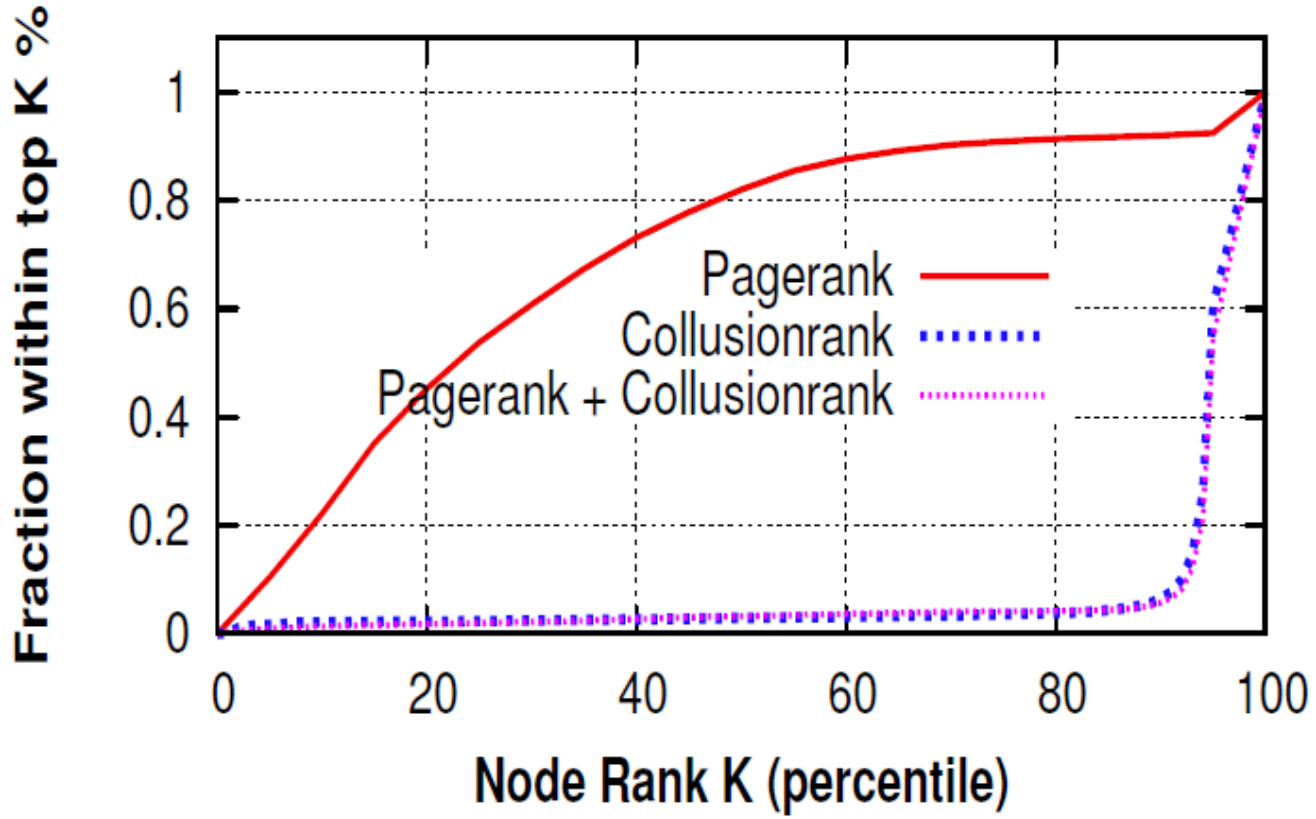
/* compute Collus
$c \leftarrow d$
while c not conve
for all nodes n

$tmp \leftarrow$

$nbr \in$

$c(n) \leftarrow \alpha \times$

end for
end while
return c



(a) Rankings of all 41,352 spammers