CS3245

# Information Retrieval

12

Lecture 12: Crawling and Link Analysis

Live Q&A
https://pollev.com/jin

# Last Time

Chapter 11

1. Classic Probabilistic Approaches for IR

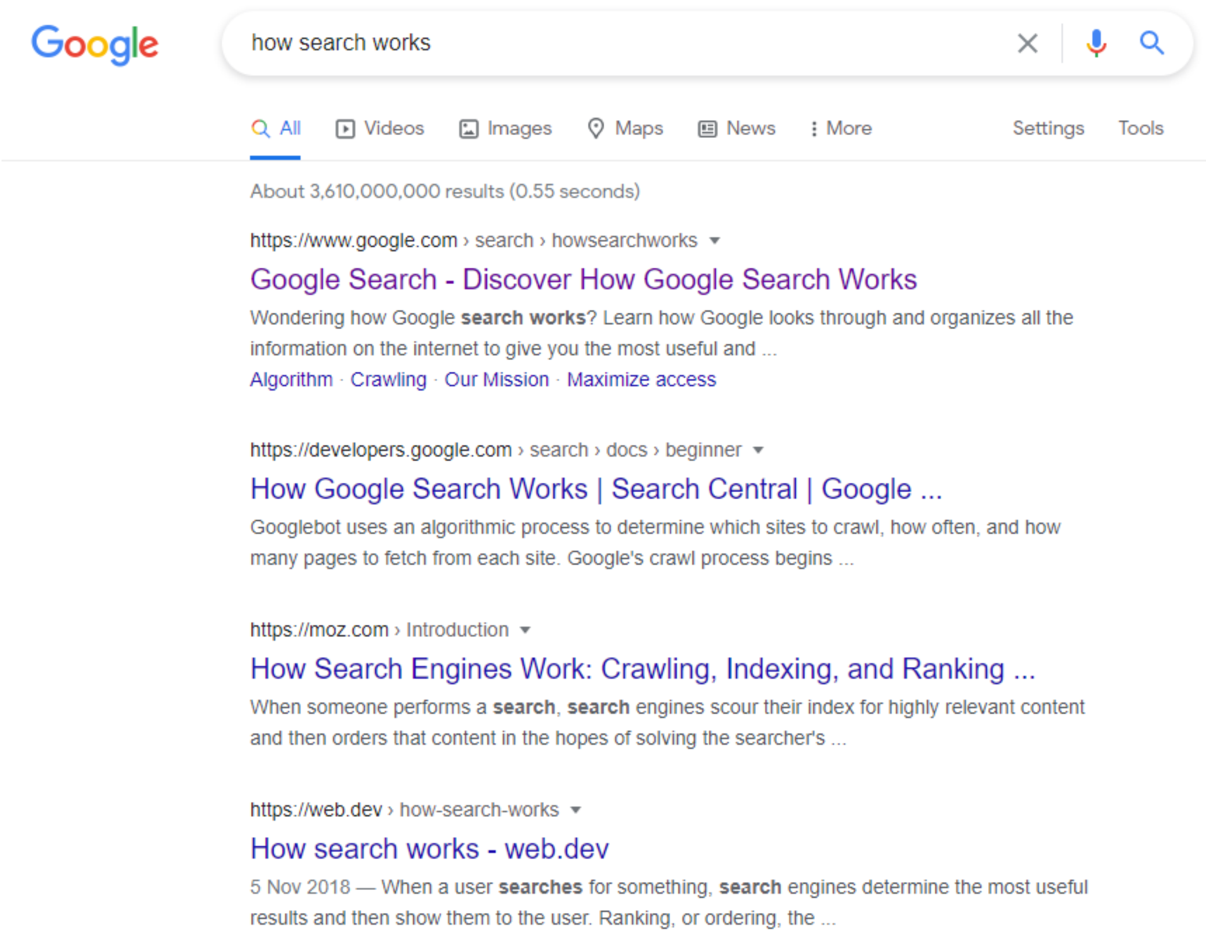
Chapter 12

1. Language Models for IR

# Today – The Web

## Chapter 20
- Crawling

## Chapter 21
- Link Analysis



# CRAWLING

# 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
- Bandwidth, latency…
  - Not just on crawler side, **but also on the web server side**.

# How hard can crawling be?

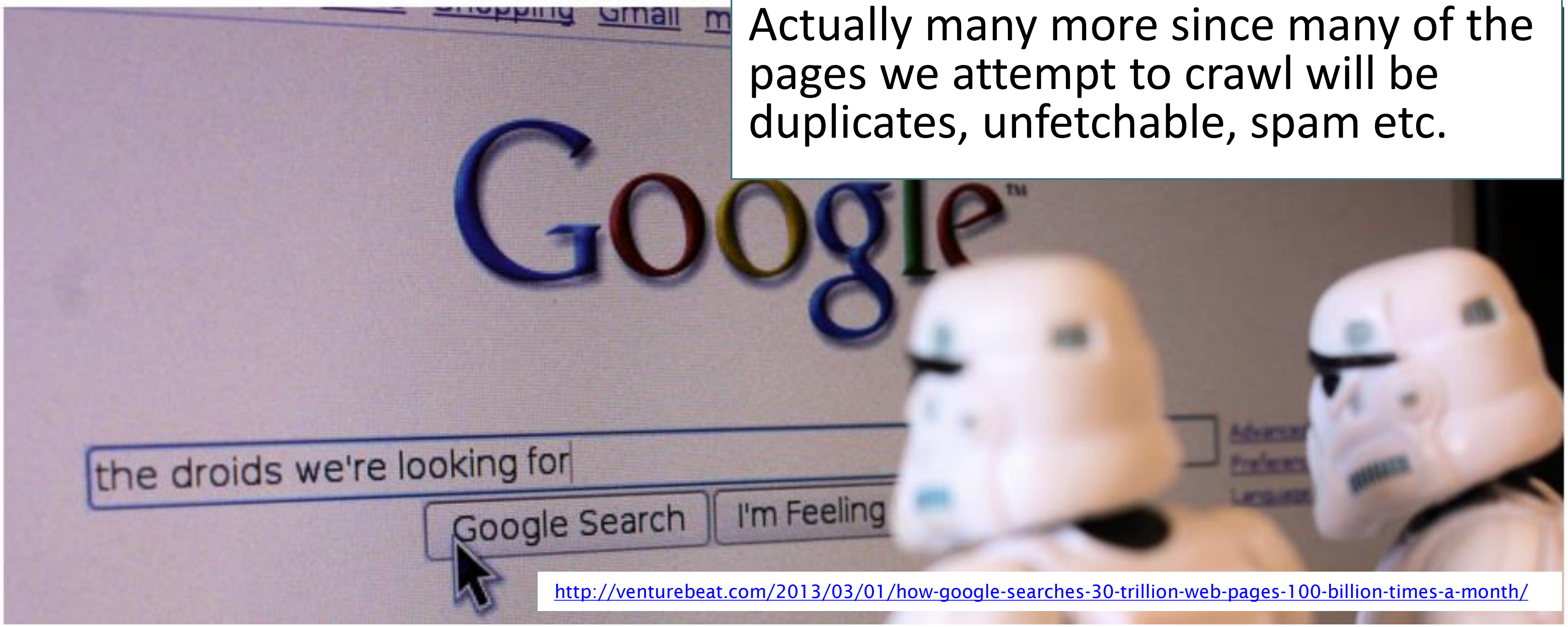
## How Google searches 30 trillion web pages, 100 billion times a month

JOHN KOETSIER, TUNE MARCH 1, 2013 12:43 PM

TAGS: FEATURED, GOOGLE, INDEX, SEARCH ENGINE, WEB SEARCH

To fetch 30T pages in one month, we need to fetch almost 11M pages per second!

Actually many more since many of the pages we attempt to crawl will be duplicates, unfetchable, spam etc.




http://venturebeat.com/2013/03/01/how-google-searches-30-trillion-web-pages-100-billion-times-a-month/

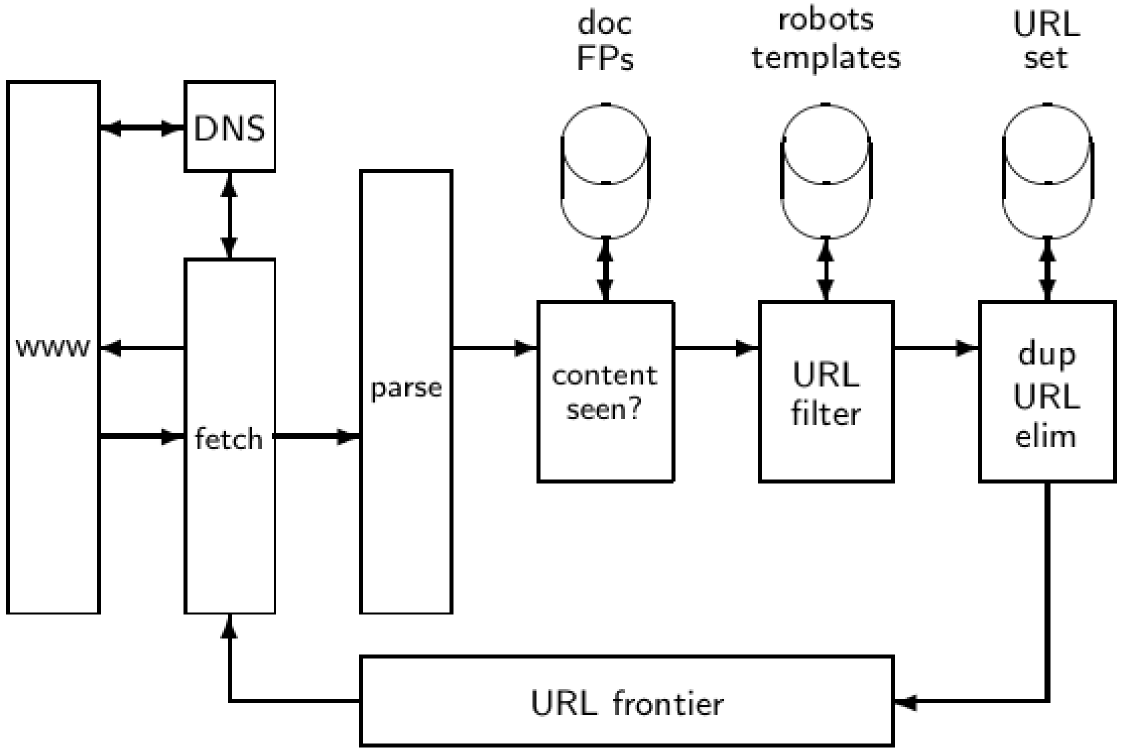
# Basic crawler operation

- Initialize queue with URLs of known seed pages
  - {https://en.wikipedia.org/, https://nlp.stanford.edu/IR-book/, …}
- Repeat
  - **Take URL from queue** — https://en.wikipedia.org/
  - **Fetch and parse page**

  - **Extract URLs from page** — https://en.wikipedia.org/wiki/Main_Page, …
  - **Add URLs to queue** — {https://nlp.stanford.edu/IR-book/, …, https://en.wikipedia.org/wiki/Main_Page, …}
  - Call the indexer to index the page
- What's wrong with this Crawler?

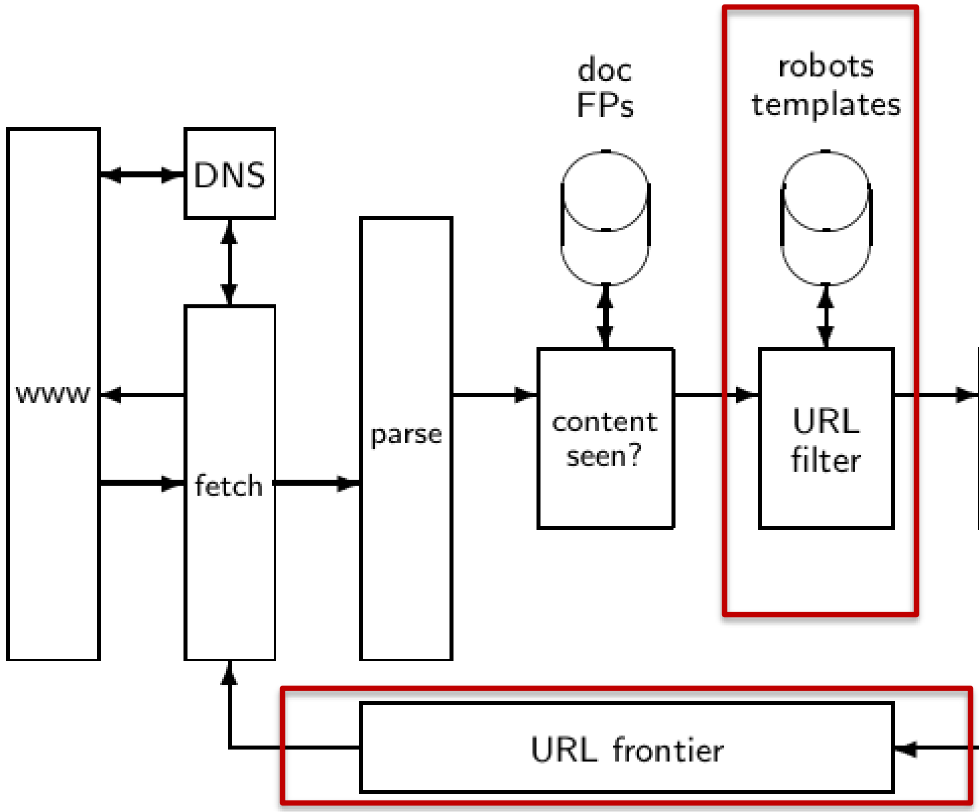
# A better architecture for Crawlers

- Fetcher / Parser / URL frontier (the queue)
- And many other components…

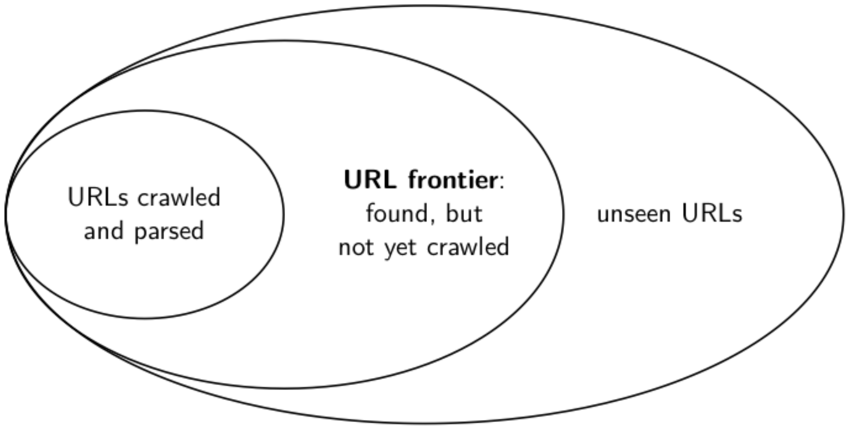


# Politeness

- Space out the requests for a site
- Crawl only the allowed pages



# URL Frontier




- The URL frontier is the data structure that holds and manages URLs we've seen, but not crawled yet.
- May include multiple pages from the same host but must **avoid trying to fetch them all at the same time**

# Robots.txt

- Protocol for giving crawlers ("robots") limited access to a website, originally from 1994
- Example:

# Observed spamming large amounts of
    https://en.wikipedia.org/?curid=NNNNNN

User-agent: MJ12bot

Disallow: /

# Wikipedia work bots:

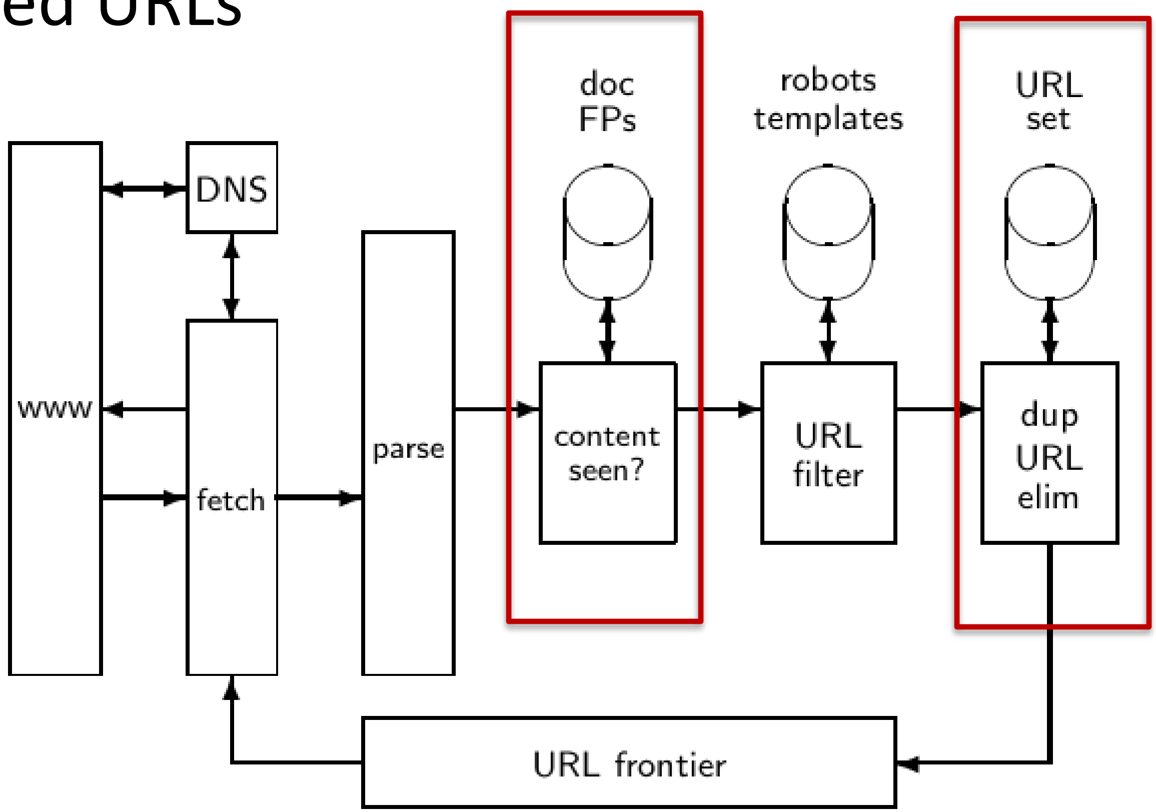
User-agent: IsraBot

Disallow:

**Important**: cache the robots.txt file of each site we are crawling

# Duplicate detection

- Duplicated contents
- Duplicated URLs



# Content seen

- For each page fetched: check if the content is already in the index
  - E.g., copied contents
- Check this using document fingerprints or shingles
- Skip documents whose content has already been indexed

# URL seen

- Duplicate elimination
  - Ignore the URLs which have been seen before.
- Normalization
  - We need to normalize (expand) all relative URLs.
    - E.g., at http://mit.edu, we may have aboutsite.html which is in fact http://mit.edu/aboutsite.html
- **Freshness**
  - Crawl some pages (e.g., news sites) more often than others

# Scalability

- Run multiple crawl threads
- Use different (geographically distributed) nodes

### Data center locations

We own and operate data centers around the world to keep our products running 24 hours a day, 7 days a week. Find out more about our data center locations, community involvement, and job opportunities in our locations around the world.

**Americas**

Berkeley County, South Carolina
Council Bluffs, Iowa
Douglas County, Georgia
Jackson County, Alabama
Lenoir, North Carolina
Mayes County, Oklahoma
Montgomery County, Tennessee
Quilicura, Chile
The Dalles, Oregon

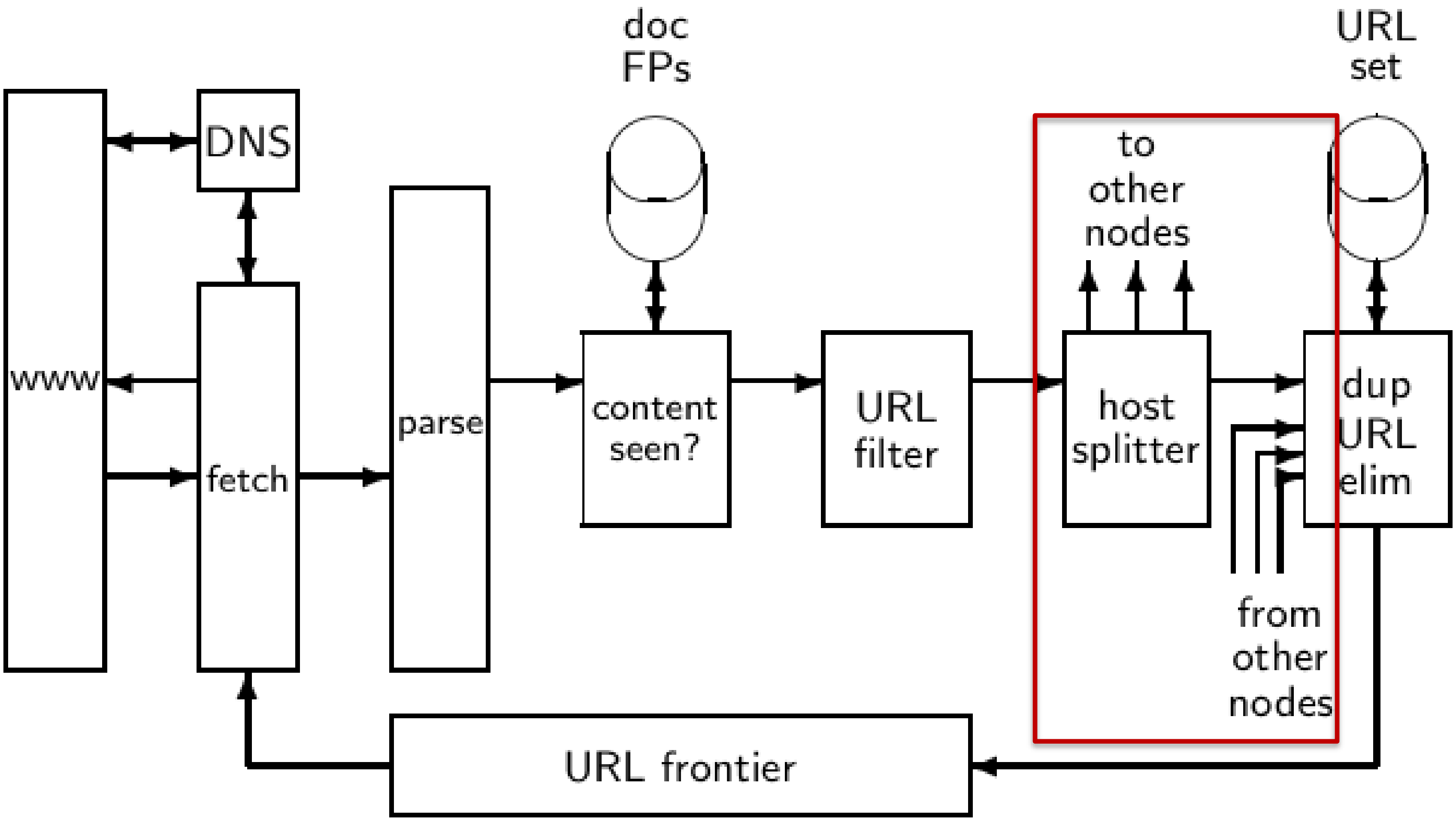
**Asia**

Changhua County, Taiwan
Singapore

**Europe**

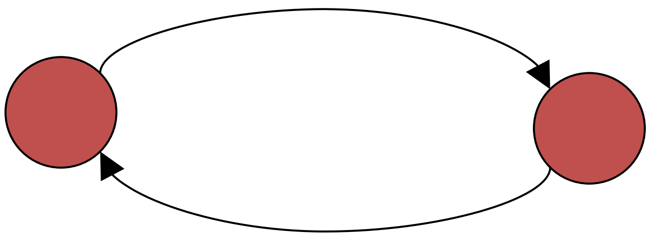
Dublin, Ireland
Eemshaven, Netherlands
Hamina, Finland
St Ghislain, Belgium

# Distributed crawling architecture



# Spider traps

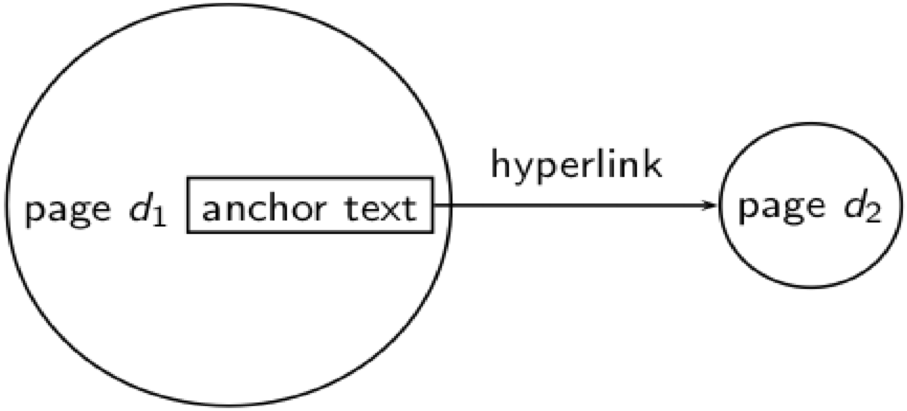
- A set of webpages that cause the crawler to go into an infinite loop or crash
  - A simple loop back



  - Calender page with a dynamic link to the next month
- May be created intentionally or unintentionally

# LINK ANALYSIS (ANCHOR TEXT)

# Anchor Text




Example: "You can find cheap cars <a href="http://...">here</a>."
Anchor text: "You can find cheap cars here."

- Assumption: The anchor text describes the content of $d_2$.
  - Anchor text is loosely defined as the text **surrounding** the hyperlink.

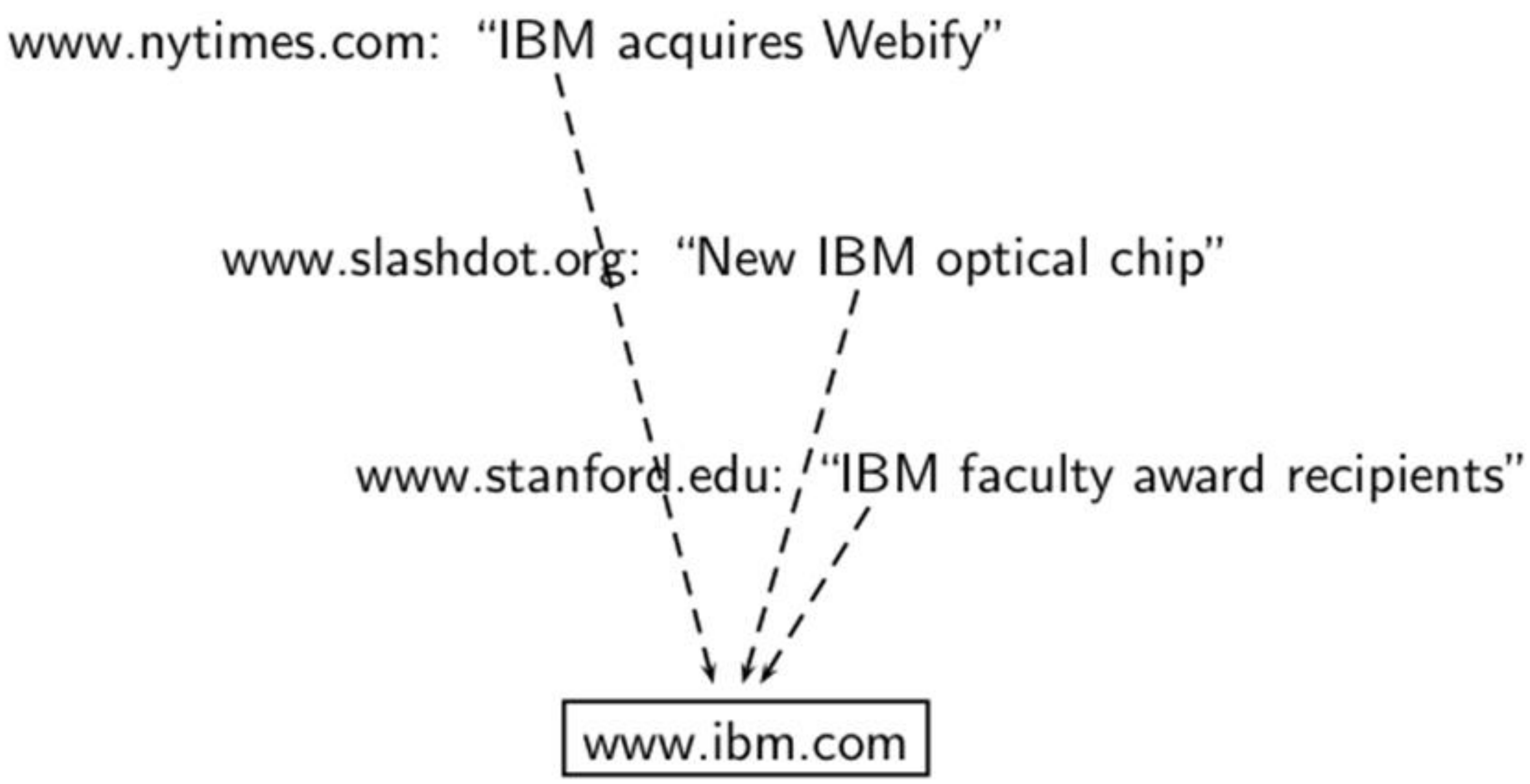
# [text of $d_2$] only vs. [text of $d_2$] + [anchor text → $d_2$]

- Searching on [text of $d_2$] + [anchor text → $d_2$] is often more effective than searching on [text of $d_2$] only.

- Example: Query *IBM*
  - IBM home page: only **a small number of mentions** of IBM and only **a few links back** to ww.IBM.com
  - Spam pages: **tons of** mentions of IBM but **without links** to the actual www.IBM.com intentionally
  - IBM Wikipedia article and many other valid pages: **a lot of** mentions of IBM **with links** to www.IBM.com

- Searching on [anchor text → $d_2$] is better for the query IBM
  - In this representation, www.IBM.com will be ranked higher…

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



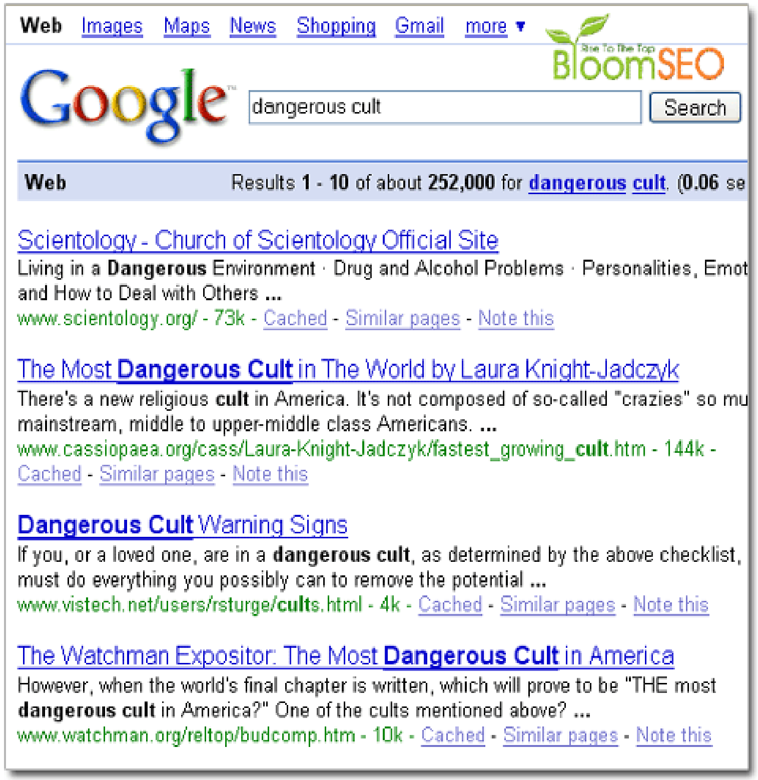
# Indexing anchor text

- 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.

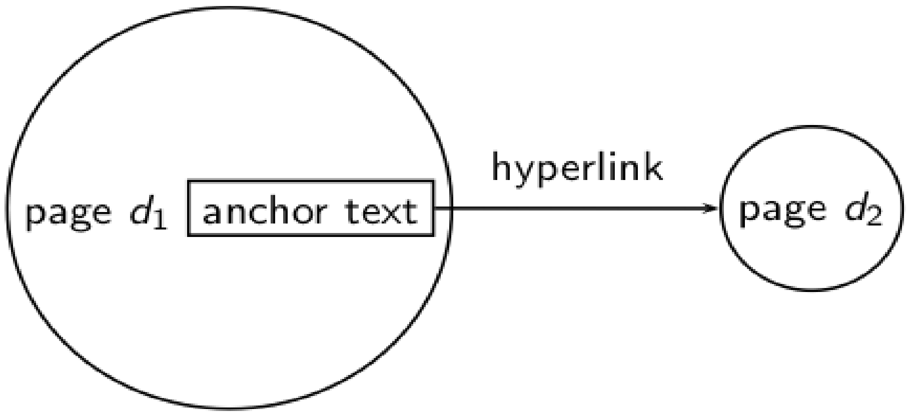
# Google bombs

- Is a search with "bad" results due to maliciously manipulated anchor text.
- E.g., [dangerous cult] on Google, Bing, Yahoo
  - Coordinated link creation by those who dislike the Church of Scientology
- Google introduced a new weighting function in January 2007 that fixed many Google bombs.
- Defused Google bombs: [who is a failure?], [evil empire]



# LINK ANALYSIS (PAGERANK)

# Quality Signal



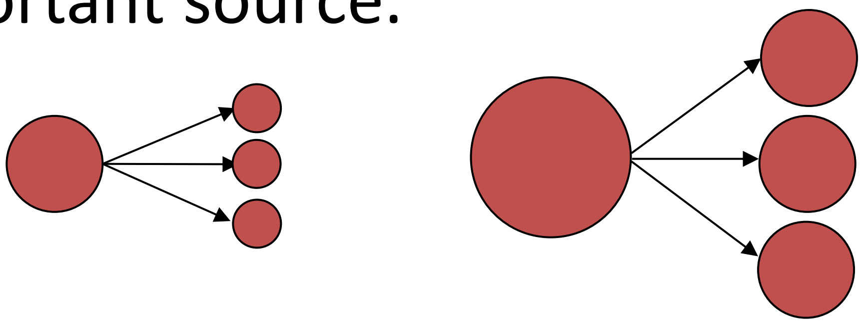

Example: "You can find cheap cars <a href="http://...">here</a>."
Anchor text: "You can find cheap cars here."

- Assumption: A hyperlink is a quality signal.
  - The hyperlink $d_1 \rightarrow d_2$ indicates that $d_1$'s author deems $d_2$ high-quality and relevant.

# Quality Signal

- It is good to have more endorsements.

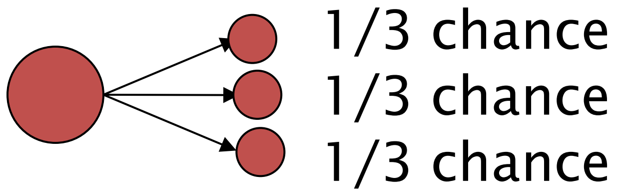
- It is good for the endorsement to come from an important source.



- It is good for the endorsement to be exclusive.

# PageRank

- Imagine a browser doing a random walk on web pages:

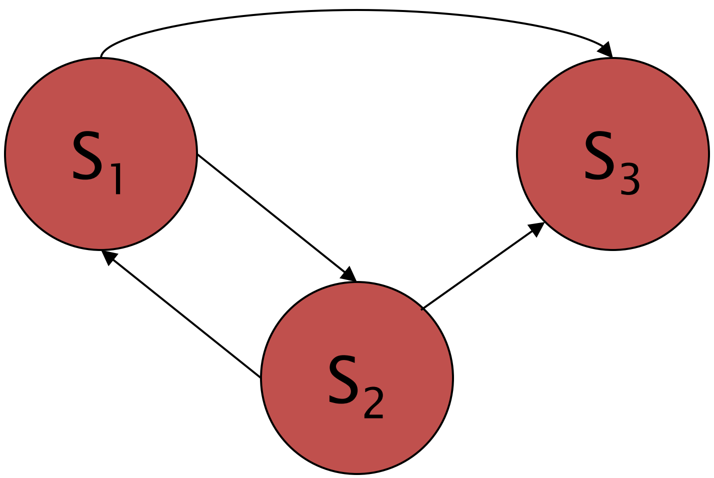



  - Start at a random page
  - At each step, follow one of the *n* links on that page, each with *1/n* probability

- Do this repeatedly. Use the "long-term visit rate" as the page's score
  - This is a global score for the page, based on the topology of the network
  - Think of it as *g(d)* from Chapter 7

# Random walks

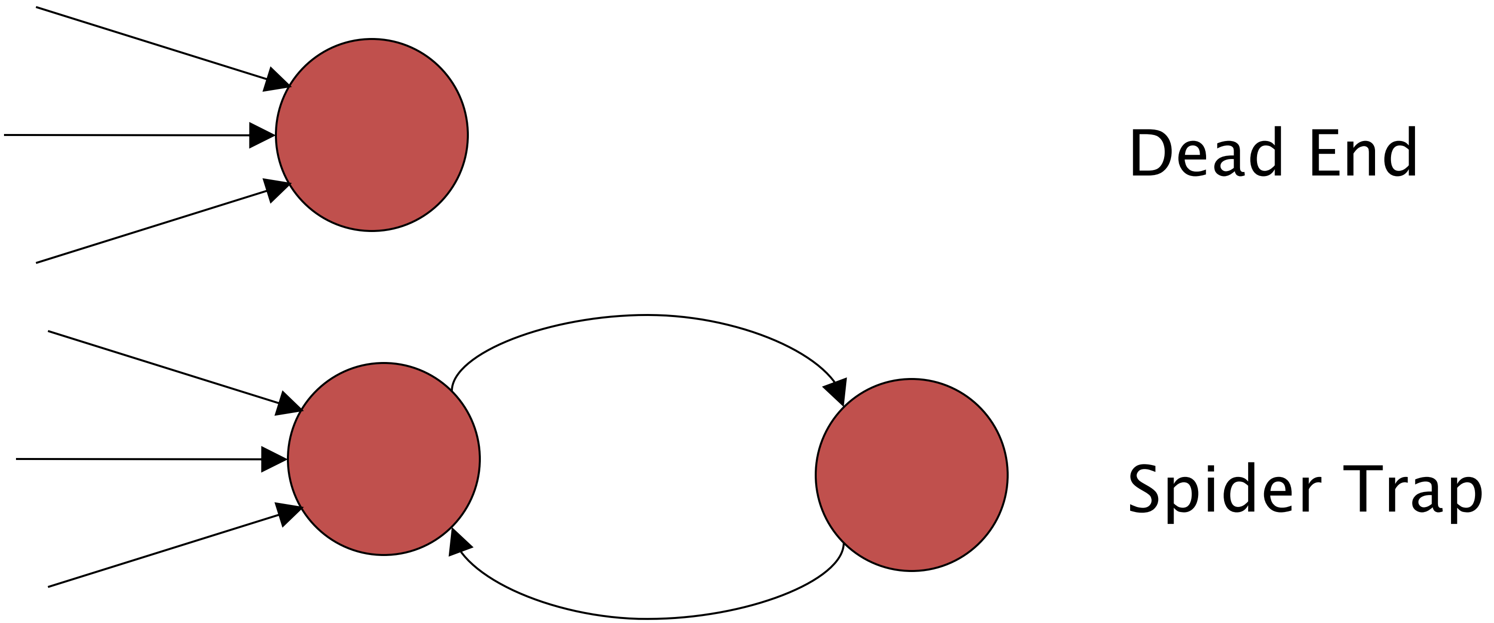
- A first order Markov chain consisting of *n* <u>states</u> and an *nxn* <u>transition probability matrix</u> *A*.
  - Each state correspond to a web page.
  - $A_{ik}$ is the probability of going from state *i* to state *k.*
  - The next state depends only on the current state.




| | $S_1$ | $S_2$ | $S_3$ |
|---|---|---|---|
| $S_1$ | 0 | 1/2 | 1/2 |
| $S_2$ | 1/2 | 0 | 1/2 |
| $S_3$ | ?? | ?? | ?? |

# Not quite enough

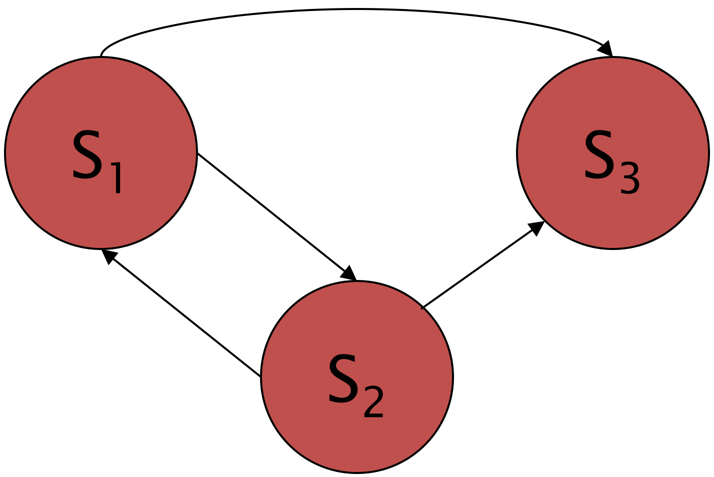
- The web is full of dead ends.
  - What sites have dead ends?
  - Our random walk can get stuck.



# Teleporting

- When a node has no outlinks
  - Teleport to a random web page
- Otherwise, at each step
  - With probability $\alpha$ (e.g., 10%), teleport to a random web page
  - With remaining probability (e.g., 90%), follow a random link on the page with equal probability

# Random walks with teleportation

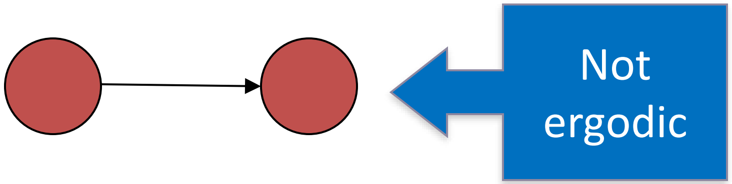



Matrix A with 10% chance of teleportation.

| | $S_1$ | $S_2$ | $S_3$ |
|---|---|---|---|
| $S_1$ | 1/30 | 29/60 | 29/60 |
| $S_2$ | 29/60 | 1/30 | 29/60 |
| $S_3$ | 1/3 | 1/3 | 1/3 |

# Ergodic Markov chains

- A Markov chain is **ergodic** if
  - you have a path from any state to any other
  - you can be in any state at every time step, with non-zero probability




  - With teleportation, our Markov chain is ergodic

- Theorem: With an ergodic Markov chain, there is a **stable** long term visit rate.

# Probability vectors

- A probability (row) vector $x = (x_1, \dots, x_n)$ tells us where the surfer is at a particular point of time
- E.g., $(0_1 00 \dots 1_i \dots, 000_n)$ → We're in state *i* with 100% probability

- More generally, the vector $x = (x_1, \dots, x_n)$ 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 $x = (x_1, \ldots, x_n)$ at this step, what is it at the next step?
  - Recall at row *i* of the transition prob. Matrix *A* tells us where we go next from state *i*.
  - So from *x*, our next state is distributed as *xA*.

$S_1$ to $S_1$ $S_2$ to $S_1$ $S_3$ to $S_1$
1*1/6 + 0*5/12 + 0*1/6

$$\vec{x}_0 = (1\ 0\ 0)$$
$$\vec{x}_1 = \vec{x}_0 \boldsymbol{A} = \begin{pmatrix} 1/6 & 2/3 & 1/6 \end{pmatrix}$$
$$\vec{x}_2 = \vec{x}_1 \boldsymbol{A} = \begin{pmatrix} 1/3 & 1/3 & 1/3 \end{pmatrix}$$

$\boldsymbol{A} =$

| | S1 | S2 | S3 |
|---|---|---|---|
| S1 | 1/6 | 2/3 | 1/6 |
| S2 | 5/12 | 1/6 | 5/12 |
| S3 | 1/6 | 2/3 | 1/6 |

# Steady State

- For any ergodic Markov chain, there is a unique long-term visit rate for each state
  - Over a long period, we'll visit each state in proportion to this rate
  - It doesn't matter where we start
    - (1, 0, 0), (0, 0, 1), (0.33, 0.33, 0.34), (0.5, 0.25, 0.25) … all these give the same long term visit rate.

# PageRank algorithm

- Regardless of where we start, we eventually reach the steady state *a*
  1. Start with any distribution (say $x = (1\ 0 \ldots 0)$).
  2. After one step, we're at $xA$.
  3. After two steps at $xA^2$, then $xA^3$ and so on.
  4. "Eventually" means for "large" k, $xA^k = a$.

- Algorithm: multiply x by increasing powers of A until the product looks stable.

# At the steady state *a*…

$$aA = a$$

- So the rank vector is an eigenvector of the adjacency matrix
  - In fact, it's the first or principal eigenvector, with corresponding eigenvalue 1.

# PageRank summary

- Pre-processing:
  - Given a graph of links, build matrix *A*
  - From it compute *a*
  - The page rank $a_i$ is a scaled number between 0 and 1.
- Query processing
  - Retrieve pages meeting query
  - Rank them by their PageRank
  - Order is *query-independent*

# 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, phrases, proximity, tiered indexes ...
  - PageRank in its original form (as presented here) 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.

# Summary

- Crawling – Obtaining documents for indexing
  - Need to be polite
- PageRank – A *g(d)* for asymmetrically linked documents

- Chapters 20 and 21 of IIR
- Resources
  - Paper on Mercator Crawler by Heydon et al.
  - Robot Exclusion Standard

> PageRank reflects our view of the importance of web pages by considering more than 500 million variables and 2 billion terms. Pages that believe are important pages receive a higher PageRank and are more likely to appear at the top of the search results"

# EXAM MATTERS

# General Information

- Date/Time: 6 May (Tue), 9-11am
- Venue: TBA
- Format
  - Pen-and-paper
  - Open book (only printed / written materials allowed)
  - Calculators (without pre-stored programs) allowed

# Exam Scope

- All lectures, tutorials and course materials (including the corresponding sections in the textbook)
- For sample questions, refer to tutorials, and past year exam papers.
- If in doubt, ask on the forum

# Types of Questions

- Q1 is true/false questions on various topics (12 marks). No justifications required.
- Q2 is short questions on various topics (44 marks). No need to show work for calculations.
- Q3-6 are long questions, topic-specific (44 marks in total). Need to show work / give justifications as required.

# Help Session

- Date/Time: 2 May (Fri), 12-2pm.
- Via Zoom
  - Recorded on a best effort basis.
- Agenda
  - Exam Paper 23/24 AY Semester 2
  - Q&A

# WHERE TO GO FROM HERE

# Learning Objectives

- You have learnt how to build your own search engine!
- In addition, you have picked up skills that are useful for your future
  - **Python** – one of the easiest and more straightforward programming languages to use.
  - **NLTK** – A good set of routines and data that are useful in dealing with NLP and IR.


IR Theory
Natural Language Processing
Distributed IR
Computational Advertising
Geographic IR
Digital Libraries
Adversarial IR
Query Analysis
Question Answering
Social Network IR
Prob IR
Recommenda tion Systems
VSM
Boolean IR
Photo credits: http://www.flickr.com/photos/aperezdc/

# Opportunities in IR

NUS
National University of Singapore

# Keep Searching

## Searches
1 UEFA Euro 2024
2 US Elections 2024
3 Olympics 2024
4 Taylor Swift
5 CDC Vouchers

## Singapore News
1 Taylor Swift
2 CDC Vouchers
3 Labubu
4 iPhone 16
5 Excessive heat

## International News
1 US Elections 2024
2 Olympics 2024
3 India Elections Results
4 Japan earthquake
5 Iran

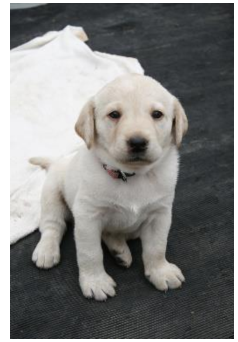
## TV Shows
1 Queen of Tears
2 Marry My Husband
3 The Double (墨雨云间)
4 Lovely Runner
5 3 Body Problem

## Movies
1 How to make millions before Grandma dies
2 Inside Out 2
3 Deadpool & Wolverine
4 Dune 2
5 Kung Fu Panda 4

## International Personalities
1 Taylor Swift
2 Donald Trump
3 Kate Middleton
4 Pope Francis
5 Kamala Harris

https://about.google/stories/year-in-search/









Thanks for joining us for the journey!