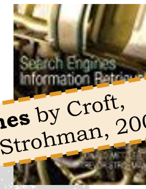


Search Engines by Croft, Metzler, and Strohman, 2009

Large-Scale Programming and Testing

Fall 2017 – CSCI 4963/6963 – Week 05

David Goldschmidt – goldschmidt@gmail.com

Office: Amos Eaton 115

Office hours: Mon/Thu 1:00-1:50PM; Wed 1:00-2:50PM

Politeness policies

- Web crawlers adhere to politeness policies
  - In general, a crawler sends a **GET** request to a specific webserver every few seconds (or minutes)
  - A website might have a **robots.txt** file, which specifies what crawlers can and cannot crawl
  - Also may specify one or more sitemaps...

Example **robots.txt** file

```
User-agent: *
Disallow: /private/
Disallow: /confidential/
Disallow: /other/
Allow: /other/public

User-agent: FavoredCrawler
Disallow:

Sitemap: http://xyz.com/sitemap.xml.gz
```

Sitemaps

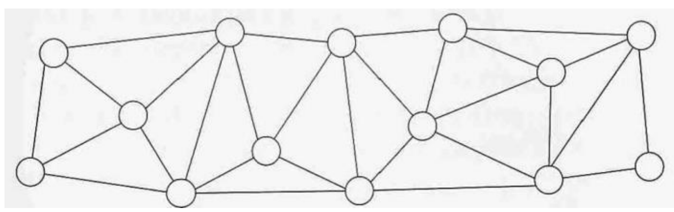
- Default priority is set at 0.5
- Some URLs might not be easily discovered by crawlers…

Example **sitemap.xml** file

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>http://www.company.com/</loc>
    <lastmod>2008-01-15</lastmod>
    <changefreq>monthly</changefreq>
    <priority>0.7</priority>
  </url>
  <url>
    <loc>http://www.company.com/inventory?item=iphone</loc>
    <changefreq>weekly</changefreq>
  </url>
  <url>
    <loc>http://www.company.com/news/</loc>
    <changefreq>daily</changefreq>
  </url>
  ...
</urlset>
```

Are you connected?

- The Internet (1969) is a network that is:
  - Global
  - Decentralized
  - Redundant
  - Heterarchical
  - Ever-changing
  - Made up of many smaller networks, which in turn are made up of varying numbers of machines
  - Made up of many different types of machines



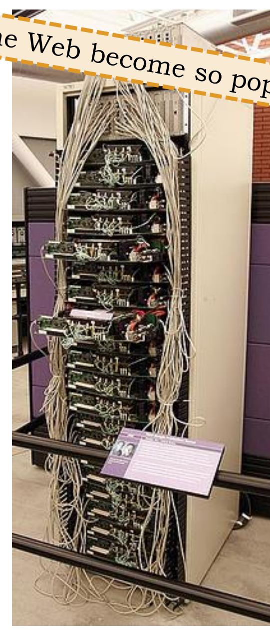
Weaving the Web

- The World Wide Web (1989) is:
  - Global
  - Decentralized
  - Redundant (sometimes)
  - Heterarchical
  - Ever-changing
  - Made up of many websites, which in turn are composed of varying numbers of webpages
  - Made up of many different types of webpages (static versus dynamic)

Who invented the Web?

Why did the Web become so popular?

- Sir Tim Berners-Lee at CERN



Why are links so important?

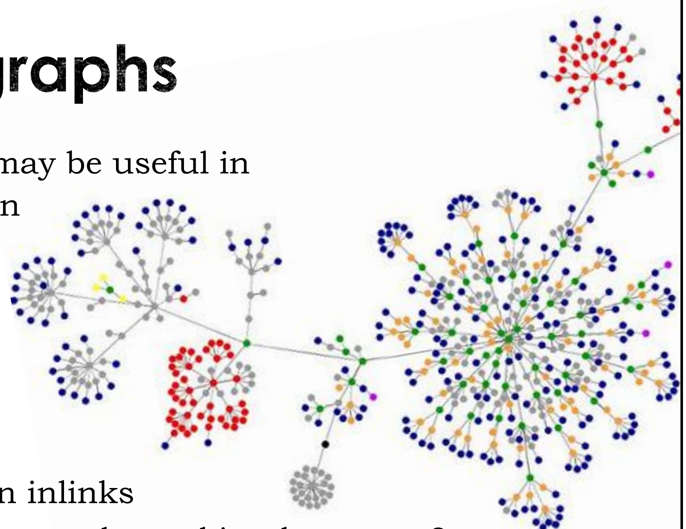
- Links (i.e., URLs) are useful to us humans for navigating websites and finding things

- Links are also extremely useful to search engines

```
<a href="http://nytimes.com">the latest news</a>
```

- Anchor (or link) text helps with ranking
  - Link text summarizes the content of the destination page
  - Link text is succinct, descriptive, and often coincides with query text
  - Link text is often written by a non-biased third party (less spam?)

Links and webgraphs



- The actual links themselves may be useful in describing a target webpage in terms of the target's:
  - Popularity
  - Importance
  - Authority
  - Incoming link (inlink) count
- Link analysis often focuses on inlinks
  - How can we use inlinks to better rank matching documents?
  - The challenge here is how we actually obtain the inlinks

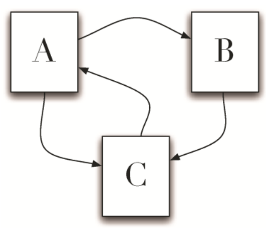
PageRank

- PageRank is an iterative link analysis algorithm
  - Use “the link structure…to calculate a quality ranking for each web page”

  - Original Google paper by the original Google guys: http://infolab.stanford.edu/~backrub/google.html

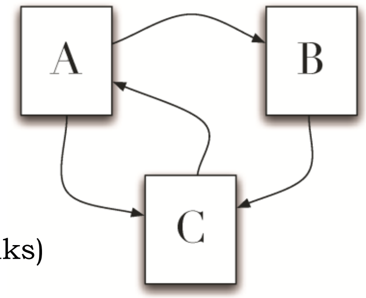
How does PageRank work?

- The PageRank of page A (i.e., PR(A)) is the probability that a “random surfer” visits that page



- Simulate browsing the Web as a random surfer:
  1. Set constant λ
  2. Choose a random number *r* between 0 and 1
  3. If *r* < λ then go to a random page
  4. Otherwise, follow a random link from the current page
  5. Go back to Step 2
- Navigating to a random page avoids getting stuck in pages with no links, with broken links, or that form cycles

PageRank



- PageRank of page C is the probability that a random surfer is viewing page C:
  - Based on both inlinks and outgoing links (outlinks)
  - C(A) is the number of outlinks from page A
  - PR(C) = PR(A)/C(A) + PR(B)/C(B)
- We assume PR forms an even distribution across all pages
  - Initially, PR(A) = PR(B) = PR(C) = 0.333
  - Then, PR(C) = 0.333/2 + 0.333/1 = 0.500
  - And PR(B) = PR(A)/C(A) = 0.333/2 = 0.166
  - And PR(A) = PR(C)/C(C) = 0.500/1 =0.500

Keep going until convergence!

PageRank

- For each page u: $$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L_v}$$
  - Here, B_u is the set of pages that point to page u
  - And, L_v is the number of deduplicated outlinks from page v
- We can account for the "random jumps" by incorporating constant λ into the equation (and N is number of pages):

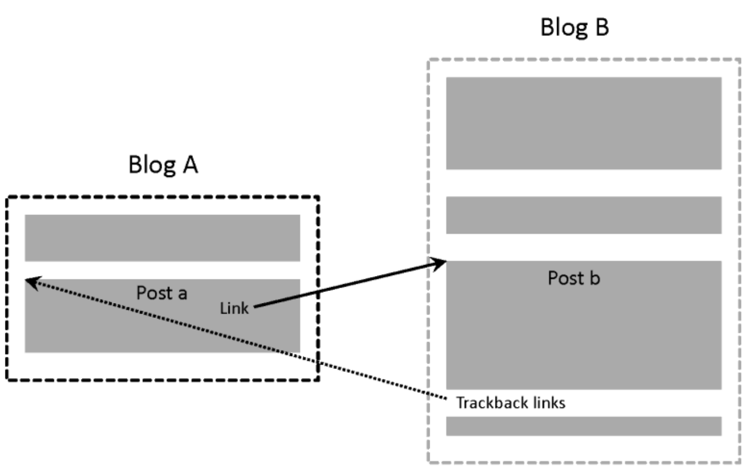
$$PR(u) = \frac{\lambda}{N} + (1 - \lambda) \sum_{v \in B_u} \frac{PR(v)}{L_v}$$

Typically, λ is low, e.g., 0.15

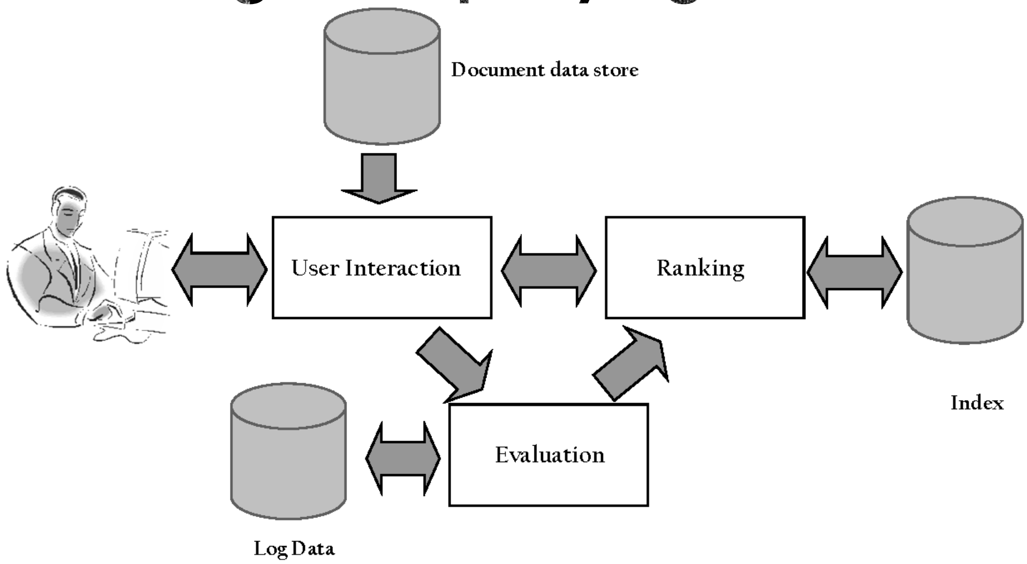
```
procedure PageRank(G)
    ▷ G is the web graph, consisting of vertices (pages) and edges (links).
    (P, L) ← G                                  ▷ Split graph into pages and links
    I ← a vector of length |P|                  ▷ The current PageRank estimate
    R ← a vector of length |P|    ▷ The resulting better PageRank estimate
    for all entries I_i ∈ I do
        I_i ← 1/|P|                     ▷ Start with each page being equally likely
    end for
    while R has not converged do
        for all entries R_i ∈ R do
            R_i ← λ/|P|  ▷ Each page has a λ/|P| chance of random selection
        end for
        for all pages p ∈ P do
            Q ← the set of pages p such that (p, q) ∈ L and q ∈ P
            if |Q| > 0 then
                for all pages q ∈ Q do
                    R_q ← R_q + (1 − λ)I_p/|Q|       ▷ Probability I_p of being a page p
                end for
            else
                for all pages q ∈ P do
                    R_p ← R_q + (1 − λ)I_p/|P|
                end for
            end if
            I ← R                         ▷ Update our current PageRank estimate
        end for
    end while
    return R
end procedure
```

Link quality and avoiding spam

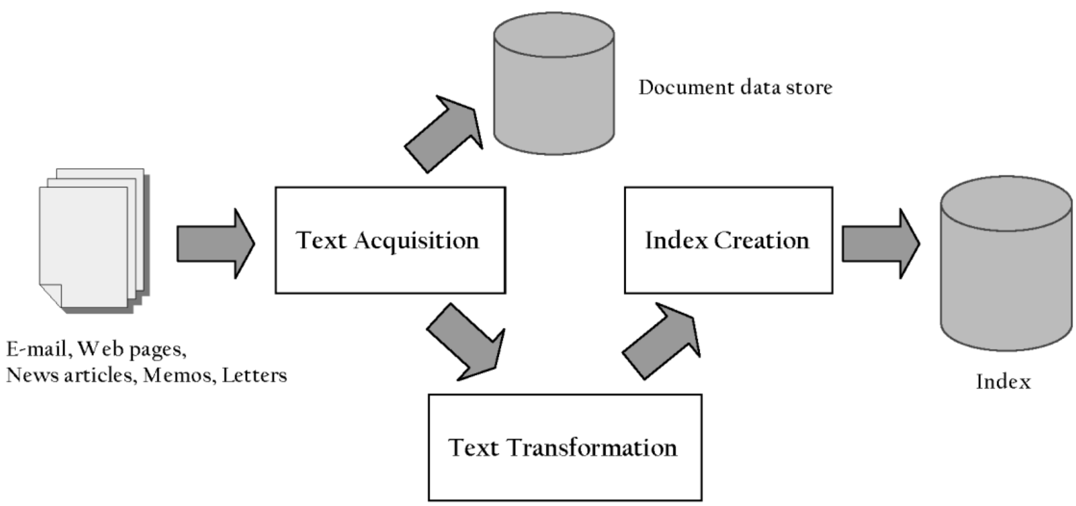
- A cycle tends to negate the effectiveness of the PR algorithm



Ranking and querying



Acquisition and indexing

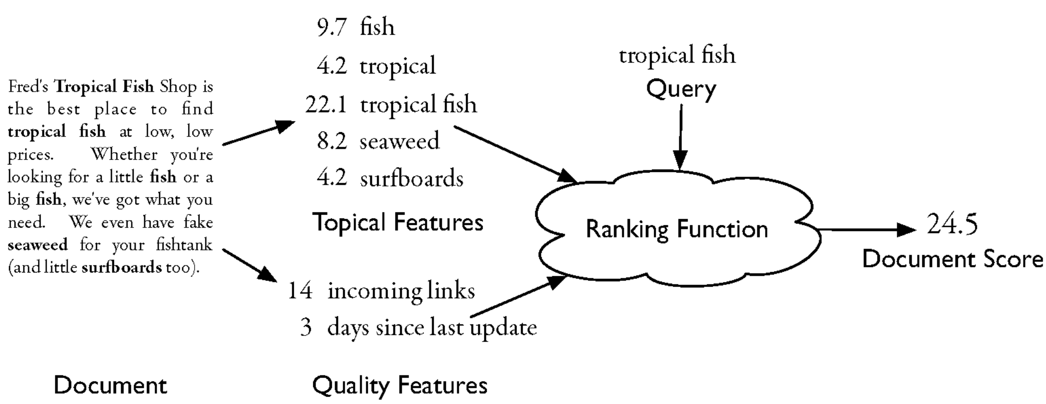


Document features

- Document features are obtained during text transformation
  - A document feature is some extractable characteristic of the document expressed numerically
  - For example, a topical feature estimates the degree to which the document is about a particular topic
  - As another example, quality features include inlink counts, the number of days since a page was last updated, etc.
  - Document features can translate into index terms

Abstract model of ranking

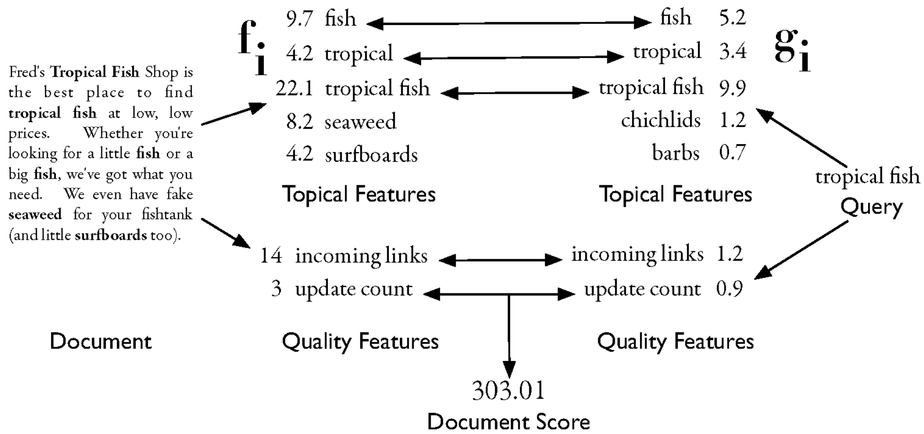
- Regardless of the ranking function, the abstract model below provides an overview of implementation



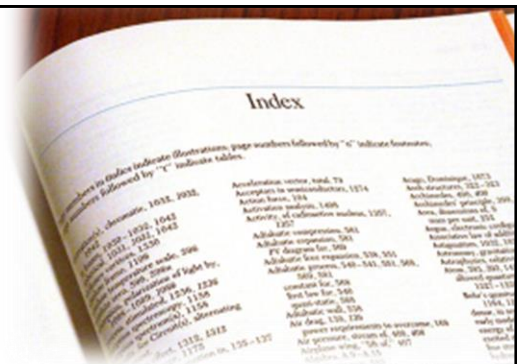
A more concrete ranking model

$$R(Q,D) = \sum_i g_i(Q) f_i(D)$$

f_i is a document feature function
g_i is a query feature function



Indexing



- An index is a data structure designed to make search very fast and efficient
- Text-based search typically requires an inverted index
  - The index is inverted because we associate documents with words rather than identifying words within or as part of a document
  - Each index term is associated with an inverted list that contains:
    - A list of documents
    - A list of word occurrences within each document
    - Word counts and positional information
    - Metadata identifying semantics

Inverted indexes

- Each entry in an inverted index is called a posting
  - The part of the posting that refers to a specific document or location is called a pointer
  - Further, each document within the collection is given a unique document number
  - Lists are usually document-ordered
  - Sorted by document number

Example document collection

- Assume each sentence below is a separate document

S_1 Tropical fish include fish found in tropical environments around the world, including both freshwater and salt water species.

S_2 Fishkeepers often use the term tropical fish to refer only those requiring fresh water, with saltwater tropical fish referred to as marine fish.

S_3 Tropical fish are popular aquarium fish, due to their often bright coloration.

S_4 In freshwater fish, this coloration typically derives from iridescence, while salt water fish are generally pigmented.

Example

- Inverted index for given documents S_1, S_2, S_3, and S_4
- Individual word occurrences have been deduplicated and ignored
- What does this structure tell us?
- What is missing?

and	1
aquarium	3
are	3 4
around	1
as	2
both	1
bright	3
coloration	3 4
derives	4
due	3
environments	1
fish	1 2 3 4
fishkeepers	2
found	1
fresh	2
freshwater	1 4
from	4
generally	4
in	1 4
include	1
including	1
iridescence	4
marine	2
often	2 3

only	2
pigmented	4
popular	3
refer	2
referred	2
requiring	2
salt	1 4
saltwater	2
species	1
term	2
the	1 2
their	3
this	4
those	2
to	2 3
tropical	1 2 3
typically	4
use	2
water	1 2 4
while	4
with	2
world	1

Example

- Inverted index for given documents S_1, S_2, S_3, and S_4, with word counts
- What does this structure tell us?
- What is missing?

and	1:1
aquarium	3:1
are	3:1 4:1
around	1:1
as	2:1
both	1:1
bright	3:1
coloration	3:1 4:1
derives	4:1
due	3:1
environments	1:1
fish	1:2 2:3 3:2 4:2
fishkeepers	2:1
found	1:1
fresh	2:1
freshwater	1:1 4:1
from	4:1
generally	4:1
in	1:1 4:1
include	1:1
including	1:1
iridescence	4:1
marine	2:1
often	2:1 3:1

only	2:1
pigmented	4:1
popular	3:1
refer	2:1
referred	2:1
requiring	2:1
salt	1:1 4:1
saltwater	2:1
species	1:1
term	2:1
the	1:1 2:1
their	3:1
this	4:1
those	2:1
to	2:2 3:1
tropical	1:2 2:2 3:1
typically	4:1
use	2:1
water	1:1 2:1 4:1
while	4:1
with	2:1
world	1:1

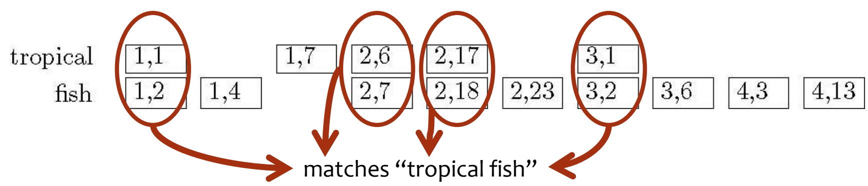
Example

- Inverted index for given documents S_1, S_2, S_3, and S_4, with specific word positions
- What does this structure tell us?
- What is missing?

Word	Postings
and	1,15
aquarium	3,5
are	3,3 · 4,14
around	1,9
as	2,21
both	1,13
bright	3,11
coloration	3,12 · 4,5
derives	4,7
due	3,7
environments	1,8
fish	1,2 · 1,4 · 2,7 · 2,18 · 2,23 · 3,2 · 3,6 · 4,3 · 4,13
fishkeepers	2,1
found	1,5
fresh	2,13
freshwater	1,14 · 4,2
from	4,8
generally	4,15
in	1,6 · 4,1
include	1,3
including	1,12
iridescence	4,9
marine	2,22
often	2,2 · 3,10
only	2,10
pigmented	4,16
popular	3,4
refer	2,9
referred	2,19
requiring	2,12
salt	1,16 · 4,11
saltwater	2,16
species	1,18
term	2,5
the	1,10 · 2,4
their	3,9
this	4,4
those	2,11
to	2,8 · 2,20 · 3,8
tropical	1,1 · 1,7 · 2,6 · 2,17 · 3,1
typically	4,6
use	2,3
water	1,17 · 2,14 · 4,12
while	4,10
with	2,15
world	1,11

Proximity matching

- Proximity matching is a technique used to identify and match multiword phrases



- Proximity matching is also used to match words within a given window of size n
  - e.g., words within five words of "fish" (n=5)

Document fields

- A document field is a section of a document that has identifiable additional semantic meaning
  - e.g., date, from:, to:, etc.
  - e.g., title, author, copyright, publisher, isbn, etc.
- Implementation options:
  - Use separate inverted lists for each field
  - Add extra information about fields to postings
  - Use extent lists….

Extent lists

- An extent is defined as a contiguous region of a document (typically with special meaning)
  - We can represent extents using word position ranges
  - The inverted list records all extents for a given field