CS276B

Text Retrieval and Mining
Winter 2005

Lecture 7

Plan for today

- Review search engine history (slightly more technically than in the first lecture)
- Web crawling/corpus construction
  - Distributed crawling
- Connectivity servers

Evolution of search engines

- First generation – use only "on page", text data
  - Word frequency, language
  - Boolean
- Second generation – use off-page, web-specific data
  - Link (or connectivity) analysis
  - Click-through data
  - Anchor-text (How people refer to this page)
- Third generation – answer "the need behind the query"
  - Semantic analysis – what is this about?
  - Focus on user need, rather than on query
  - Context determination
  - Helping the user
    - UI, spell checking, query refinement, query suggestion, syntax driven feedback, context help, context transfer, etc
  - Integration of search and text analysis

1995-1997 AV, Excite, Lycos, etc

From 1998. Made popular by Google but everyone now

Evolving

Connectivity analysis

- Idea: mine hyperlink information of the Web
- Assumptions:
  - Links often connect related pages
  - A link between pages is a recommendation "people vote with their links"
- Classic IR work (citations = links) a.k.a. "Bibliometrics" [Kess63, Garf72, Smal73, …]. See also [Lars96].
- Much Web related research builds on this idea [Piro96, Aroc97, Sper97, Carr97, Klei98, Brin98,…]

Third generation search engine: answering "the need behind the query"

- Semantic analysis
  - Query language determination
    - Auto filtering
    - Different ranking (if query in Japanese do not return English)
  - Hard & soft matches
    - Personalities (triggered on names)
    - Cities (travel info, maps)
    - Medical info (triggered on names and/or results)
    - Stock quotes, news (triggered on stock symbol)
    - Company info …

Answering "the need behind the query"

- Context determination
  - spatial (user location/target location)
  - query stream (previous queries)
  - personal (user profile)
  - explicit (vertical search, family friendly)
- Context use
  - Result restriction
  - Ranking modulation

Spatial context – geo-search

- Geo-coding
  - Geometrical hierarchy (squares)
  - Natural hierarchy (country, state, county, city, zip-codes)
- Geo-parsing
  - Pages (infer from phone nos, zip, etc)
  - Queries (use dictionary of place names)
  - Users
    - Explicit (tell me your location)
    - From IP data
  - Mobile phones
    - In its infancy, many issues (display size, privacy, etc)

Helping the user

- UI
- Spell checking
- Query completion
- …

Crawling

Crawling Issues

- How to crawl?
  - *Quality*: "Best" pages first
  - *Efficiency*: Avoid duplication (or near duplication)
  - *Etiquette*: Robots.txt, Server load concerns
- How much to crawl? How much to index?
  - *Coverage*: How big is the Web? How much do we cover?
  - *Relative Coverage:* How much do competitors have?
- How often to crawl?
  - *Freshness*: How much has changed?
  - How much has really changed? (why is this a different question?)

Basic crawler operation

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

Simple picture – complications

- Web crawling isn't feasible with one machine
  - All of the above steps distributed
- Even non-malicious pages pose challenges
  - Latency/bandwidth to remote servers vary
  - Robots.txt stipulations
    - How "deep" should you crawl a site's URL hierarchy?
  - Site mirrors and duplicate pages
- Malicious pages
  - Spam pages (Lecture 1, plus others to be discussed)
  - Spider traps – incl dynamically generated
- Politeness – don't hit a server too often

Robots.txt

- Protocol for giving spiders ("robots") limited access to a website, originally from 1994
  - www.robotstxt.org/wc/norobots.html
- Website announces its request on what can(not) be crawled
  - For a URL, create a file `URL/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":

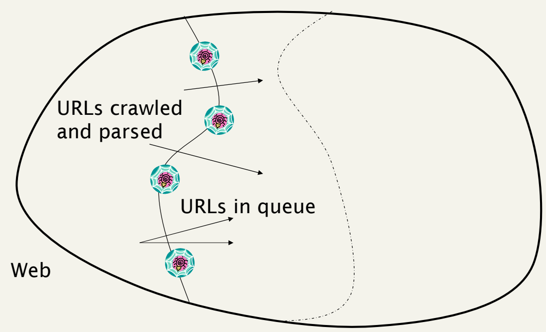
```
User-agent: *
Disallow: /yoursite/temp/

User-agent: searchengine
Disallow:
```

Crawling and Corpus Construction

- Crawl order
- Distributed crawling
- Filtering duplicates
- Mirror detection

Where do we spider next?



Crawl Order

- Want best pages first
- Potential quality measures:
  - Final In-degree
  - Final Pagerank

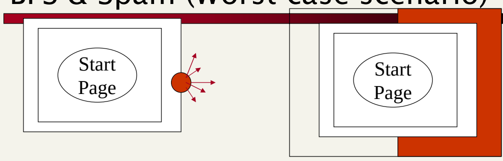
What's this?

Crawl Order

- Want best pages first
- Potential quality measures:
  - Final In-degree
  - Final Pagerank
- Crawl heuristic:
  - Breadth First Search (BFS)
  - Partial Indegree
  - Partial Pagerank
  - Random walk

Measure of page quality we'll define later in the course.

BFS & Spam (Worst case scenario)



BFS depth = 2

Normal avg outdegree = 10

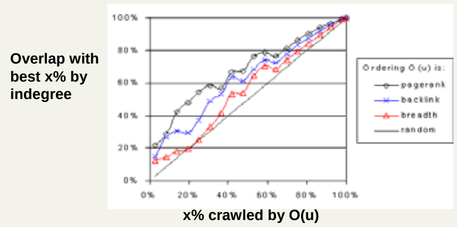
100 URLs on the queue including a spam page.

Assume the spammer is able to generate dynamic pages with 1000 outlinks

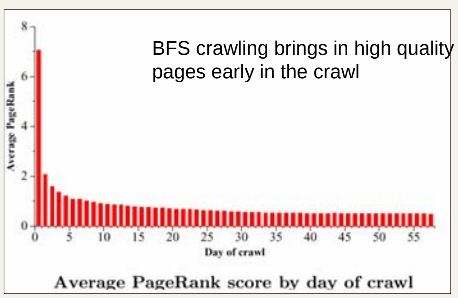
BFS depth = 3
2000 URLs on the queue
50% belong to the spammer

BFS depth = 4
1.01 million URLs on the queue
99% belong to the spammer

Stanford Web Base (179K, 1998) [Cho98]



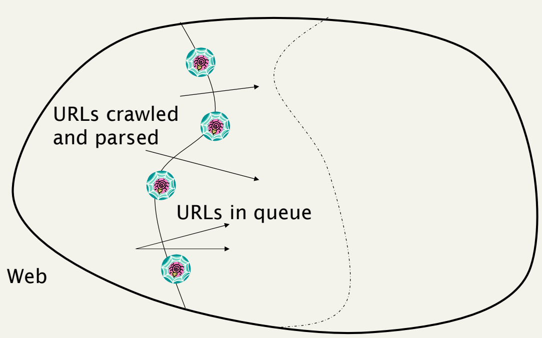
Web Wide Crawl (328M pages, 2000) [Najo01]



Queue of URLs to be fetched

- What constraints dictate which queued URL is fetched next?
- Politeness – don't hit a server too often, even from different threads of your spider
- How far into a site you've crawled already
  - Most sites, stay at ≤ 5 levels of URL hierarchy
- Which URLs are most promising for building a high-quality corpus
  - This is a graph traversal problem:
  - Given a directed graph you've partially visited, where do you visit next?

Where do we spider next?



Where do we spider next?

- Keep all spiders busy
- Keep spiders from treading on each others' toes
  - Avoid fetching duplicates repeatedly
- Respect politeness/robots.txt
- Avoid getting stuck in traps
- Detect/minimize spam
- Get the "best" pages
  - What's best?
  - Best for answering search queries

Where do we spider next?

- Complex scheduling optimization problem, subject to all the constraints listed
  - Plus operational constraints (e.g., keeping all machines load-balanced)
- Scientific study – limited to specific aspects
  - Which ones?
  - What do we measure?
- What are the compromises in distributed crawling?

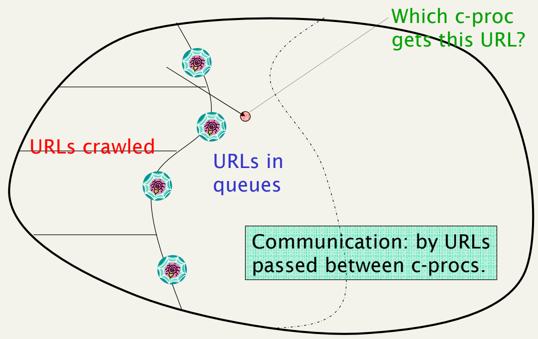
Parallel Crawlers

- We follow the treatment of Cho and Garcia-Molina:
  - http://www2002.org/CDROM/refereed/108/index.html
- Raises a number of questions in a clean setting, for further study
- Setting: we have a number of *c-proc*'s
  - c-proc = crawling process
- Goal: we wish to spider the *best* pages with minimum *overhead*
  - What do these mean?

Distributed model

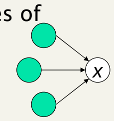
- Crawlers may be running in diverse geographies – Europe, Asia, etc.
  - Periodically update a master index
  - Incremental update so this is "cheap"
    - Compression, differential update etc.
  - Focus on communication overhead during the crawl
- Also results in dispersed WAN load

c-proc's crawling the web



Measurements

- Overlap = *(N-I)/I* where
  - *N* = number of pages fetched
  - *I* = number of distinct pages fetched
- Coverage = *I/U* where
  - *U* = Total number of web pages
- Quality = sum over downloaded pages of their *importance*
  - Importance of a page = its in-degree
- Communication overhead =
  - Number of URLs c-proc's exchange

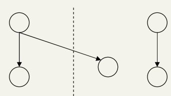


Crawler variations

- c-procs are independent
  - Fetch pages oblivious to each other.
- Static assignment
  - Web pages partitioned statically a priori, e.g., by URL hash … more to follow
- Dynamic assignment
  - Central co-ordinator splits URLs among c-procs

Static assignment

- Firewall mode: each c-proc only fetches URL within its partition – typically a domain
  - inter-partition links not followed
- Crossover mode: c-proc may following inter-partition links into another partition
  - possibility of duplicate fetching
- Exchange mode: c-procs periodically exchange URLs they discover in another partition



Experiments

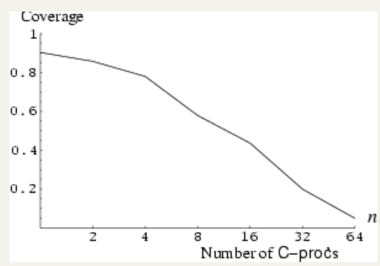
- 40M URL graph – Stanford Webbase
  - Open Directory (dmoz.org) URLs as seeds
- Should be considered a small Web

Summary of findings

- Cho/Garcia-Molina detail many findings
  - We will review some here, both qualitatively and quantitatively
  - You are expected to understand the reason behind each qualitative finding in the paper
  - You are not expected to remember quantities in their plots/studies

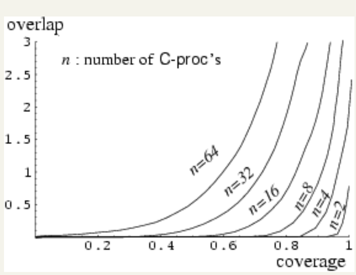
Firewall mode coverage

- The price of crawling in firewall mode



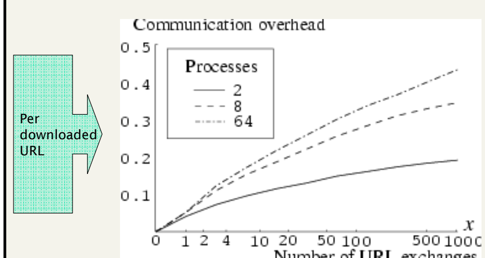
Crossover mode overlap

- Demanding coverage drives up overlap



Exchange mode communication

- Communication overhead sublinear



Connectivity servers

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

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

Stores mappings in memory from
    - URL to outlinks, URL to inlinks
- Applications
  - Crawl control
  - Web graph analysis
    - Connectivity, crawl optimization
  - Link analysis
    - More on this later

Most recent published work

- Boldi and Vigna
  - http://www2004.org/proceedings/docs/1p595.pdf
- Webgraph – set of algorithms and a java implementation
- Fundamental goal – maintain node adjacency lists in memory
  - For this, compressing the adjacency lists is the critical component

Adjacency lists

- The set of neighbors of a node
- Assume each URL represented by an integer
- Properties exploited in compression:
  - Similarity (between lists)
  - Locality (many links from a page go to "nearby" pages)
  - Use gap encodings in sorted lists
    - As we did for postings in CS276A
  - Distribution of gap values

Storage

- Boldi/Vigna report getting down to an average of ~3 bits/link
  - (URL to URL edge)
  - For a 118M node web graph

Resources

- www.robotstxt.org/wc/norobots.html
- www2002.org/CDROM/refereed/108/index.html
- www2004.org/proceedings/docs/1p595.pdf