Interoperability Architectures and Techniques

CS 431 – March 6, 2006
Carl Lagoze – Cornell University

## What is the problem?

- Getting heterogeneous systems to work together
- Providing the user with a seamless information experience
- What services do you want to provide?
  - Search and access?
  - More?
    - information access
    - authorization and authentication
    - integrity and reliability
    - Reuse
- How much human intervention?
- Level of perfection?

## Why is it hard

- Differences in…
  - hardware
  - applications
  - design patterns
  - language
  - culture
  - laws
  - policies
  - human behaviors

## Interoperability is multidimensional

- Syntax
  - XML
- Semantics
  - RDF/RDFS/OWL
- Vocabularies/Ontologies
  - Dublin Core/ABC/CIDOC-CRM
- Search and discovery
  - Z39.50
  - SDLIP
  - ZING
- Protocols
  - Dienst
  - OAI-PMH
- Information models
  - METS
  - FEDORA
  - DIDL

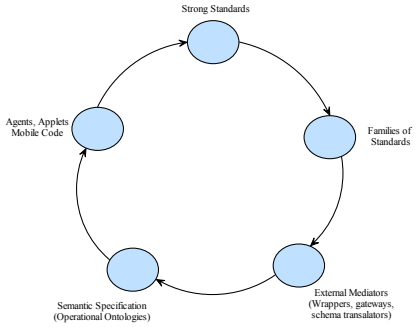
## Contrast to Distributed Systems

- Distributed systems
  - Collections of components at different sites that are carefully designed to work with each other
- Heterogeneous or federated systems
  - Cooperating systems in which individual components are designed or operated autonomously

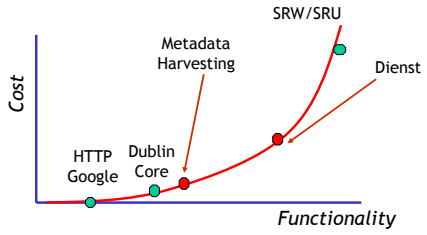
## Measuring success of interoperability solutions

- Degree of component autonomy
- Cost of infrastructure
- Ease of contributing components
- Ease of using components
- Breadth of task complexity supported by the solution
- Scalability in the number of components

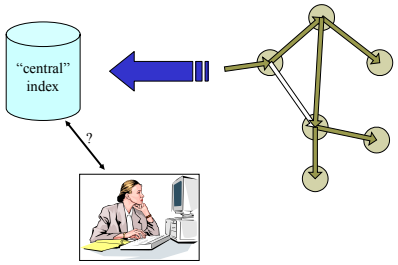
## Families of interoperability solutions




## Interoperability Trade-offs




## Web Search Strategies – Crawling and Automated Indexing




## Definition

Spider = robot = crawler

Crawlers are computer programs that roam the Web with the goal of automating specific tasks related to the Web.

## Crawlers and internet history

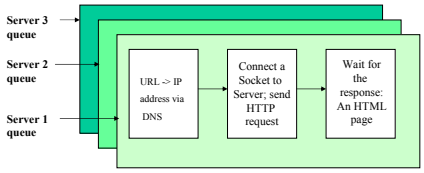
- 1991: HTTP
- 1992: 26 servers
- 1993: 60+ servers; self-register; archie
- 1994 (early) – first crawlers
- 1996 – search engines abound
- 1998 – focused crawling
- 1999 – web graph studies
- 2002 – use for digital libraries (focused crawling)

## So, why not write a robot?

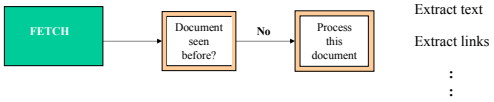
You'd think a crawler would be easy to write:

```
Pick up the next URL
Connect to the server
GET the URL
When the page arrives, get its links
(optionally do other stuff)
REPEAT
```

## The Central Crawler Function



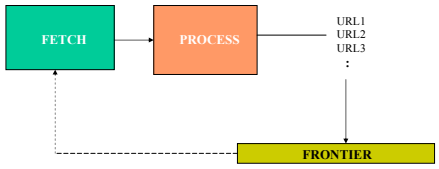

## Handling the HTTP Response




## LINK Extraction

- Finding the links is easy (sequential scan)
- Need to clean them up and canonicalize them
- Need to filter them
- Need to check for robot exclusion
- Need to check for duplicates

## Update the Frontier




## Crawler Issues

- System Considerations
- The URL itself
- Politeness
- Visit Order
- Robot Traps
- The hidden web

## Standard for Robot Exclusion

- Martin Koster (1994)
- http://any-server:80/robots.txt
- Maintained by the webmaster
- Forbid access to pages, directories
- Commonly excluded: /cgi-bin/
- Adherence is voluntary for the crawler

## Visit Order

- The frontier
- Breadth-first: FIFO queue
- Depth-first: LIFO queue
- Best-first: Priority queue
- Random
- Refresh rate

## Robot Traps

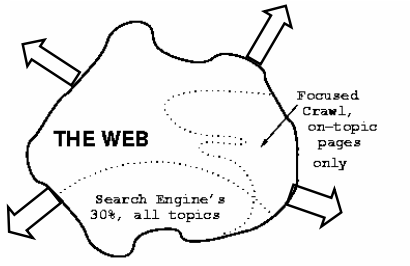
- Cycles in the Web graph
- Infinite links on a page
- Traps set out by the Webmaster

## The Hidden Web

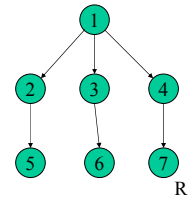
- Dynamic pages increasing
- Subscription pages
- Username and password pages

- Research in progress on how crawlers can "get into" the hidden web

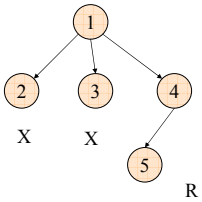
## Focused Crawling




## Focused Crawling




Breadth-first crawl




Focused crawl



## Focusing the Crawl

- Threshold: page is on-topic if correlation to the closest centroid is above this value
- Cutoff: follow links from pages whose "distance" from closest on-topic ancestor is less than this value

## Automated Information Discovery

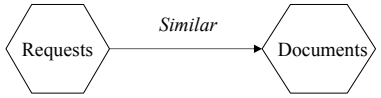
Creating metadata records manually is labor-intensive and hence expensive.

The aim of **automated information discovery** is for users to discover information without using skilled human effort to build indexes.

## Similarity Ranking

**Ranking methods using similarity**

- Measure the degree of similarity between a query and a document (or between two documents).
- Basic technique is the ***vector space model*** with term weighting.




**Similar:** How similar is document to a request?

## Vector Space Methods: Concept

*n*-dimensional space, where *n* is the total number of different terms (words) in a set of documents.

Each document is represented by a vector, with magnitude in each dimension equal to the (weighted) number of times that the corresponding term appears in the document.

Similarity between two documents is the angle between their vectors.

*Much of this work was carried out by Gerald Salton and colleagues in Cornell's computer science department.*

## Example 1: Incidence Array

terms in $d_1$ -> ant ant bee
terms in $d_2$ -> bee hog ant dog
terms in $d_3$ -> cat gnu dog eel fox

| terms | ant | bee | cat | dog | eel | fox | gnu | hog |
|---|---|---|---|---|---|---|---|---|
| $d_1$ | 1 | 1 | | | | | | |
| $d_2$ | 1 | 1 | | 1 | | | | 1 |
| $d_3$ | | | 1 | 1 | 1 | 1 | 1 | |

Weights: $t_{ij} = 1$ if document $i$ contains term $j$ and zero otherwise

## Reasons for Term Weighting

Similarity using an incidence matrix measures the occurrences of terms, but no other characteristics of the documents.

Terms are more useful for information retrieval if they:

- appear several times in one document (weighting by term frequency)
- only appear in some documents (weighting by document frequency)
- appear in short document (weighting by document length)

## Inverse Document Frequency

**Concept**

A term that occurs in a few documents is likely to be a better discriminator that a term that appears in most or all documents.

## Issues in extending traditional IR for the Web

- Traditional IR benchmarks were relatively small scale (large is about 20GB)
- Web queries are very short
- Quality is an issue in ranking
- Sex, lies, and the hidden web
- Polysemy due to domain overlap

- Web has context and "hints"
  - Structure of pages (e.g. html title might be rated higher)
  - Implicit metadata of link context
    - Anchor text of citing pages
    - Weighting influenced by citation structure (recall Garfield)

## PageRank Algorithm (Google)

Concept:

The rank of a web page is higher if many pages link to it.

Links from highly ranked pages are given greater weight than links from less highly ranked pages.

## PageRank with Damping Factor Intuitive Model

A user:

1. Starts at a random page on the web

2a. With probability p, selects any random page and jumps to it

2b. With probability 1-p, selects a random hyperlink from the current page and jumps to the corresponding page

3. Repeats Step 2a and 2b a very large number of times

Pages are ranked according to the relative frequency with which they are visited.

## Information Retrieval Using PageRank

Simple Method

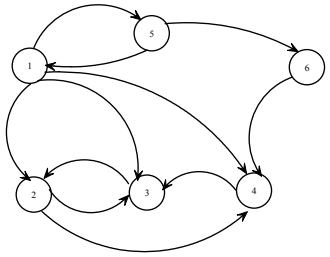
Consider all hits (i.e., all document vectors that share at least one term with the query vector) as equal.

Display the hits ranked by PageRank.

The disadvantage of this method is that it gives no attention to how closely a document matches a query

With dynamic document sets, references patterns are calculated for a set of documents that are selected based on each individual query.

## Google Example




## Adjacency Matrix

Citing page (from)

| Cited page (to) | $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_5$ | $P_6$ | Number |
|---|---|---|---|---|---|---|---|
| $P_1$ | | | | | 1 | | 1 |
| $P_2$ | 1 | | 1 | | | | 2 |
| $P_3$ | 1 | 1 | | 1 | | | 3 |
| $P_4$ | 1 | 1 | | | 1 | 1 | 4 |
| $P_5$ | 1 | | | | | | 1 |
| $P_6$ | | | | | 1 | | 1 |
| Number | 4 | 2 | 1 | 1 | 3 | 1 | |

## Normalize by Number of Links from Page

Citing page

| Cited page | $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_5$ | $P_6$ |
|---|---|---|---|---|---|---|
| $P_1$ | | | | | 0.33 | |
| $P_2$ | 0.25 | | 1 | | | |
| $P_3$ | 0.25 | 0.5 | | 1 | | |
| $P_4$ | 0.25 | 0.5 | | | 0.33 | 1 |
| $P_5$ | 0.25 | | | | | |
| $P_6$ | | | | | 0.33 | |
| Number | 4 | 2 | 1 | 1 | 3 | 1 |

$= \mathbf{B}$

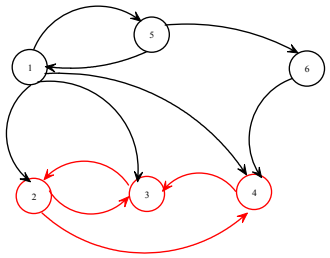
**Normalized link matrix**

## Iterate until convergence

Iterate: $\mathbf{w}_k = \mathbf{B}\mathbf{w}_{k-1}$

| $\mathbf{w}_1$ | $\mathbf{w}_2$ | $\mathbf{w}_3$ | $\mathbf{w}_4$ | ... converges to ... | $\mathbf{w}$ |
|---|---|---|---|---|---|
| 1 | 0.33 | 0.08 | 0.03 | -> | 0.00 |
| 1 | 1.25 | 1.83 | 2.80 | -> | 2.39 |
| 1 | 1.75 | 2.79 | 2.06 | -> | 2.39 |
| 1 | 2.08 | 1.12 | 1.05 | -> | 1.19 |
| 1 | 0.25 | 0.08 | 0.02 | -> | 0.00 |
| 1 | 0.33 | 0.08 | 0.03 | -> | 0.00 |

## Motivating the Damping Factor




## The PageRank Iteration

The basic method iterates using the normalized link matrix, B.

wk = Bwk-1

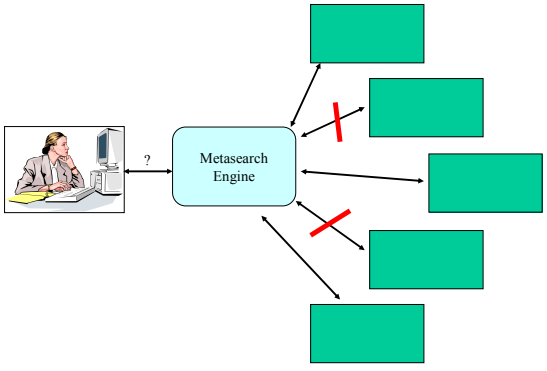
This w is the high order eigenvector of B

Google iterates using a damping factor. The method iterates using a matrix B', where:

B' = dN + (1 - d)B

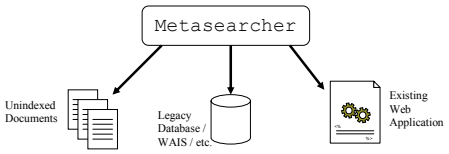
N is the matrix with every element equal to 1/n.
d is a constant found by experiment.

## Web Search Strategies - Metasearching



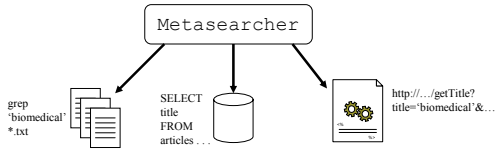

## What is "Metasearching"?

- Given many document **sources** and a **query**, a metasearcher:
  - Finds the good sources for the query
  - Evaluates the query at these sources
  - Merges the results from these sources



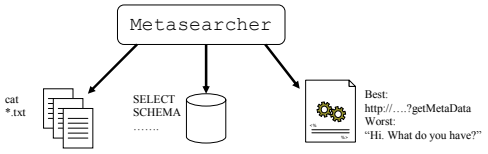
## Metasearching Issues

- How to query different types of sources?
- How to combine results and rankings from multiple data sources?




## Metasearching Issues . . . Cont'd

- How to choose among multiple data sources?
- How to get metadata about multiple data sources?




## ZING

http://www.loc.gov/z3950/agency/zing/zing-home.html

## The problem

- Search syntax differs across engines
  - http://www.google.com/search?hl=en&ie=ISO-8859-1&q=dogs+and+cats&btnG=Google+Search
  - http://search.yahoo.com/search?fr=fp-pull-web-t&p=dogs+and+cats
  - http://search.msn.com/results.aspx?FORM=MSNH&q=dogs%20and%20cats
- Means of returning results sets differs

## Aims of ZING

- Common framework (implemented as protocol) for searching over multiple servers
- Builds on notion of metadata (attribute-based access points to information.
- Components
  - •CQL – Common query syntax, keyword and attribute based
  - •SRU – REST based transmission of requests
  - •SRW – SOAP based transmission of requests

## Technical history

**Z39.50**

- Developed for X.25 networks (connection orientation), conversion to run over TCP fitted later
- Original concept in days when repeating a search was expensive computation (about 1980)

## SRW/SRU services

- Explain
  - Return information about the database – search access points (e.g, title, author) metadata formats returned
- Scan
  - Return information about an index term (e.g., related terms)
- Search
  - Return search results

## SRW Result Sets

- Server may support notion of persistent result sets
  - Return an ID of the set from query
- Client may perform operations on result sets
  - Refine searches
  - Chunk results
- Server makes "commitment" to retain result set but may change commitment.

## Dienst – Beyond just searching

- is a protocol and reference implementation of a distributed digital library service
- where a network of services provide
- World Wide Web browser access,
- uniform search over distributed indexes,
- and access to structured documents.



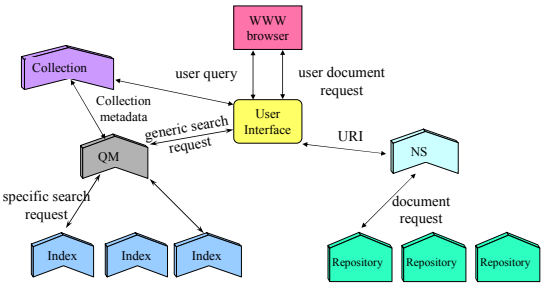
## Why a service based protocol?

- Expose the operational semantics of the services through an API,
- to permit flexible integration of the services,
- and use of the services by other clients/consumers/services.

## Defining the services

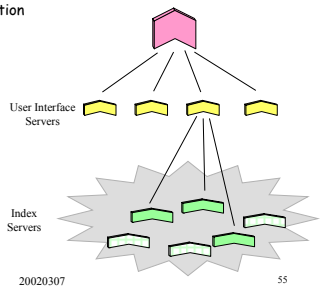
- Repository – deposit, storage, and access to structured documents.
- Index – process queries on documents and returned handles
- Query Mediator – route queries to appropriate indexes
- Collection – define services and content in logical collections
- User Interface – human-oriented front-end for services.
- Name Server – Resolves URN's (handles) to document location(s)

## Dienst Services



## Collection Service

- Periodically polled by each user interface server for
  - elements of the collection
  - index servers for the collection






## Why a Document Model?

- "Documents" in current web are both:
  - Unstructured
  - Chaotic
- Different views and pieces of contents are needed for:
  - Bandwidth reduction
  - Rights management
  - Usability

## Dienst Document Model

- *Metadata* – support for multiple descriptive formats
- *Views* – alternative expression or structural representation of the content encapsulated in the digital object
- *Divs* – hierarchically nested structure contained in a view

## Expressing the document model in the protocol

- *Structure* – expose the views and structure for the digital object
- *Disseminate* – select the structural component (and packaging of it) to disseminate
- *List-Meta-Formats* – list available descriptive formats