Searching the Web

A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke and S. Raghavan

Ali Taleghani
January 26, 2005
Top 3 results for searching “Java”

- **Google**
  1. java.sun.com
  2. www.java.com
  3. javaboutique.internet.com

- **Altavista**
  1. altova.com
  2. oracle.com
  3. vbcoffee.com (actual coffee site)
Overview

- Intro to searching the Web
- Crawling Web pages
- Storing crawled pages
- Indexing
- Ranking & link analysis
- Pitfalls of current searching
- Conclusion
- Google stats
Crawling Web Pages

- Crawlers are small programs browsing Web
- Extract URLs from Web pages
- URLs passed to *Crawler Control*
- Crawler Control determines next URLs to visit & places on queue
- Crawler gets next URL from queue
Initialize:
    UrlsDone = ∅
    UrlsTodo = {‘yahoo.com/index.htm’, ..}

Repeat:
    url = UrlsTodo.getNext()

    ip = DNSlookup( url.getHostname() )
    html = DownloadPage( ip, url.getPath() )

    UrlsDone.insert( url )

    newUrls = parseForLinks( html )
    For each newUrl
        If not UrlsDone.contains( newUrl )
            then UrlsTodo.insert( newUrl )
Challenges of Crawling

- Which pages should the crawler download?
- How should the crawler refresh pages?
- How should the load on Web sites be minimized?
- How should crawling be parallelized?
Page Selection

What is “important”?
- Interest Driven: Textual similarity
- Popularity Driven: Page backlinks
- Location Driven: Location of page P

How does the crawler operate?
- Want to visit important pages first
- Can visit fixed # of pages or fixed # of “important” pages
Page Selection

How to guess good pages to visit?

- All URLs saved in queue
- Crawler picks next URL so it has highest “value”
- Value based on importance of page & is only an estimate
Refreshing Pages

- Pages have to be refreshed to be kept up-to-date
- Two strategies for refresh
  - Uniform – all pages are refreshed
  - Proportional – changing pages visited more proportionally (estimated)
Reducing load on Web sites

- Response from sites might be slow
- Not all domains wish to be crawled (robots.txt)
- Pages should be downloaded at reasonable rate – need concurrent connections
- Google tried to crawl an online game
Crawling in parallel

- **Natural Unit of work is URL**
- **Different approaches:**
  - Google uses centralized URL server
    - Another three crawling machines
    - Communication with URL server only
  - URL space can be divided into n pieces
    - Each machine completely in charge of one piece
    - Links outside an URL space are passed to appropriate server
Page Repository has two functions:
- Interface for crawler to store pages
- Provide API for indexers to access

Challenges
- Scalability
- Dual Access Modes: random / streaming
- Large bulk updates
- Obsolete pages
Designing a Distributed Page Repository

- Page Distribution across nodes
  - Uniform distribution
  - Hash distribution

- Physical Page Organization
  - Hash buckets – pages distributed based on identifier
  - Random access supported using B-tree
Designing a Distributed Page Repository

- **Update Strategies** (generated by crawler)
  - Batch-mode / steady crawler
    - Batch-mode: Executed periodically
    - Steady: Runs without any pause
  - Partial / Complete crawls (batch-mode)
    - Partial: crawl subset of pages
    - Complete: Crawl all pages
  - Updates can be in-place or shadowing
    - In-place: Pages from crawler integrated immediately
    - Shadowing: Pages stored separately and updated later
Indexing

- Several different indexes built
  - Link index
  - Text index
  - Utility index

- Text indexes as an inverted index
  - Sorted list of locations for a term
  - Additional criteria considered (e.g. `<H1>`, `<B>`)

Index Partitioning

- Building inverted index requires scalable & distributed architecture
- Two strategies for partitioning index:
  - Local Inverted File
    - Node responsible for disjoint subset of pages
    - Query sent to all nodes, each return disjoint result
  - Global Inverted File
    - A node responsible for subset of terms
    - Query only sent to some nodes
Ranking & Link Analysis

- Web too large & unorganized
- Web pages not self descriptive
- Results of a query have to be sorted
- Sorting based on link structure
  - PageRank
  - HITS
PageRank

- Tries to capture notion of importance
- Rank of $P$ based on # of links pointing to it
- Also considered: Importance of pages pointing to $P$
- Google used PageRank first
  - Google looks at anchor text -> non-text information becomes “searchable”
HITS Hypertext-Induced Topic Search

- Uses Authority and Hub score
- Authority pages most relevant to a query
- Hub pages point to authorities
- Hubs used to calculate authority pages
- Authorities hardly point to other authorities
Impact of Search Engines on Page popularity

- Experiments work
  - Popular pages get more popular & vice-versa
- Theoretical work
  - Unpopular pages need more time to become known
  - Once known, popularity increase quickly
Pitfalls of current Searching

- Scamming Google
  - “more evil than Satan” -> microsoft.com
  - “miserable failure” -> George W. Bush

- Many links made to point to a page
  - Hubs point to authorities...

- First one no longer works
  - It appears that Google no longer indexes this page
Conclusion

- Thorough overview of major aspects of searching the web
- Major problems associated with scale, rate of change & heterogeneity of Web
- Most work related to own experience (small data)
- Difficult to know what companies do as it is secret
- Paper discusses only “known” approaches
- Published at a time when little search engine success
Google vital stats (2001)

- 6000 Linux machines
  - 33 die every day
- 500TB of disk storage
- 1 Google day = 16.5 machine years
  - \((6,000/365)\)
- 50 million queries per day
  - 1000 queries / sec
- 3 data replication centres
References

Thank You!