Modern Information Retrieval

Web cralwing and search

Hamid Beigy

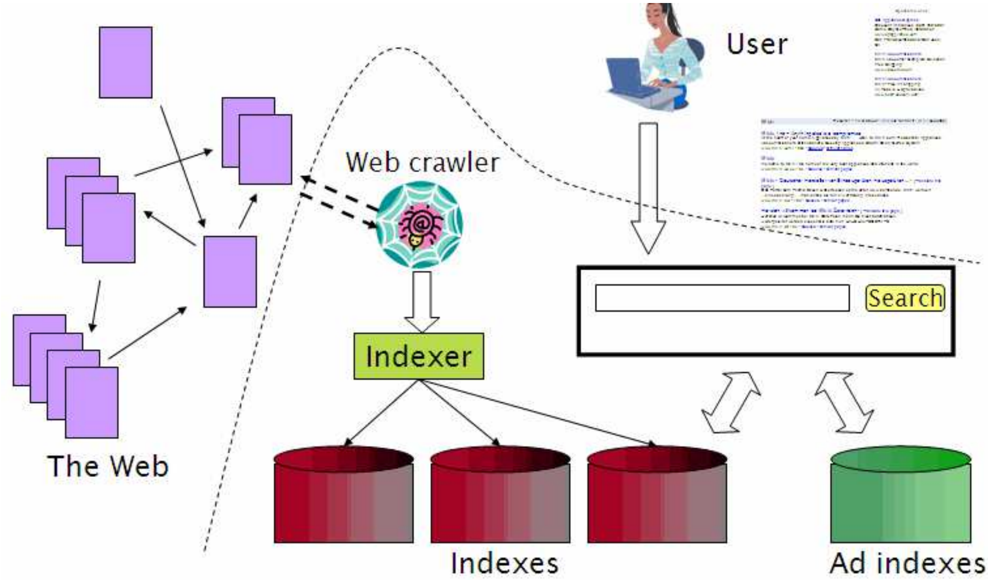
Sharif university of technology

December 25, 2022

Table of contents

Introduction

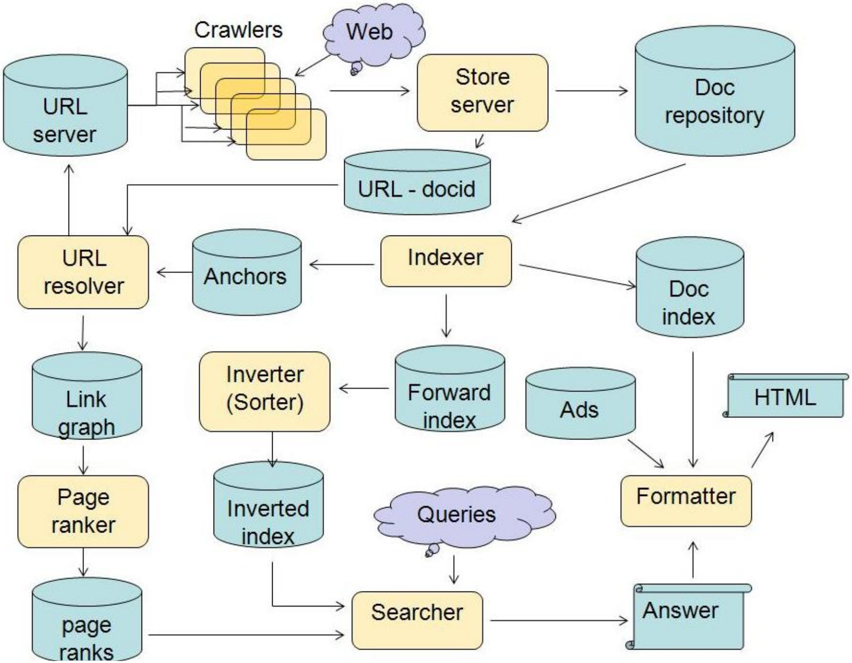
Web Search



The World Wide Web is huge.

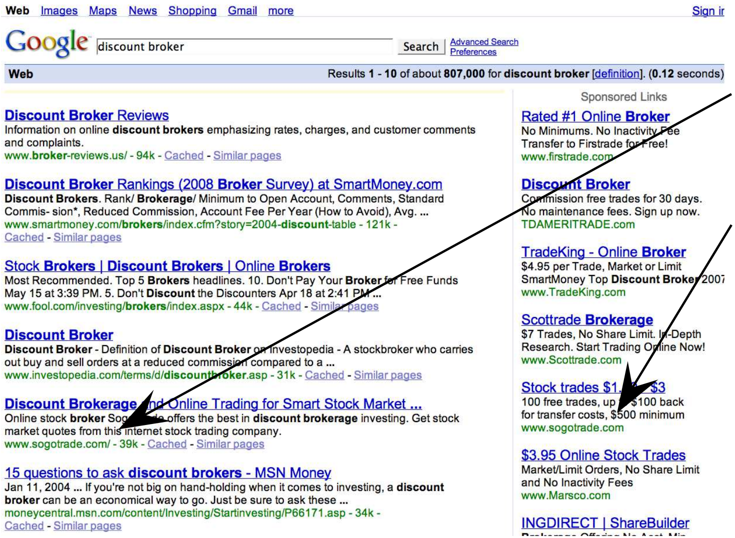
1. 100,000 indexed pages in1994.
2. 10,000,000,000's indexed pages in 2013.
3. 30 trillion pages in the Google Index in 2022 (100 million gigabytes).
4. Most queries will return millions of pages with high similarity.
5. Content(text) alone cannot discriminate.
6. Use the structure of the Web(a graph).
7. Gives indications of usefulness of each page.

1. Without search, content is hard to find.
2. Without search, there is no incentive to create content.
   - Why publish something if nobody will read it?
   - Why publish something if I don't get ad revenue from it?
3. Somebody needs to pay for the web.
   - Servers, web infrastructure, content creation
   - A large part today is paid by search ads.
   - Search pays for the web.
4. On the web, search is not just a nice feature, search is a key enabler of the web.


Crawlers
Web
URL server
Store server
Doc repository
URL - docid
URL resolver
Anchors
Indexer
Doc index
Link graph
Inverter (Sorter)
Forward index
Ads
HTML
Page ranker
Inverted index
Queries
Formatter
page ranks
Searcher
Answer

Results of a query

Web pages (left) and ads (right)



SogoTrade appears in search results.

SogoTrade appears in ads.

Do search engines rank advertisers higher than non-advertisers?

All major search engines claim no.

Duplicate detection

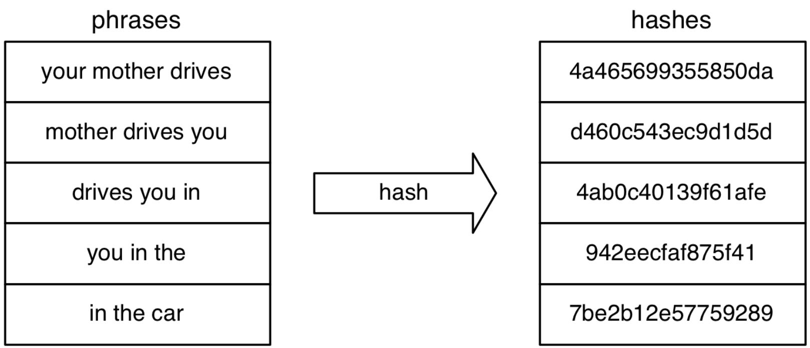
1. The web is full of duplicated content (30%–40%).
2. More so than many other collections
3. Exact duplicates (easy to eliminate by using hash/fingerprint)
4. Near-duplicates (difficult to eliminate)
5. For the user, it's annoying to get a search result with near-identical documents.
6. We need to eliminate near-duplicates.

1. Computing similarity with an edit-distance measure
2. We want syntactic (as opposed to semantic) similarity.
   True semantic similarity (similarity in content) is too difficult to compute.
3. We do not consider documents near-duplicates if they have the same content, but express it with different words.
4. Use similarity threshold θ to make the call is/isn't a near-duplicate.
   For example, two documents are near-duplicates if similarity $> \theta = 80\%$.

- A shingle is simply a word n-gram.
- Shingles are used as features to measure syntactic similarity of documents.
- For example, for $n = 3$, a rose is a rose is a rose would be represented as this set of shingles:
  { a-rose-is, rose-is-a, is-a-rose }
- Let U be a set and A and B be subsets of U, then the Jaccard coefficient is defined as

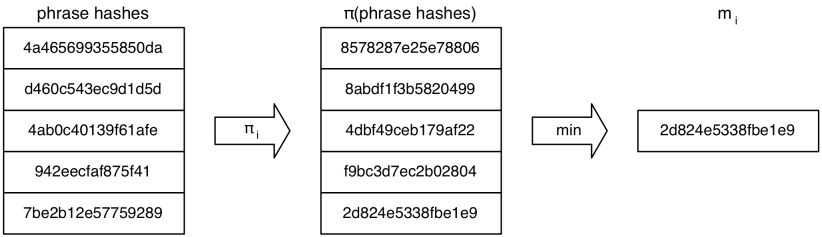
$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

- Computing the Jaccard coefficient for two documents needs high computation time.
- We define the similarity of two documents as the Jaccard coefficient of their shingle sets.
- To avoid this, we use a form of hashing.
- We map every shingle into a hash value over large space (for example 64-bits).

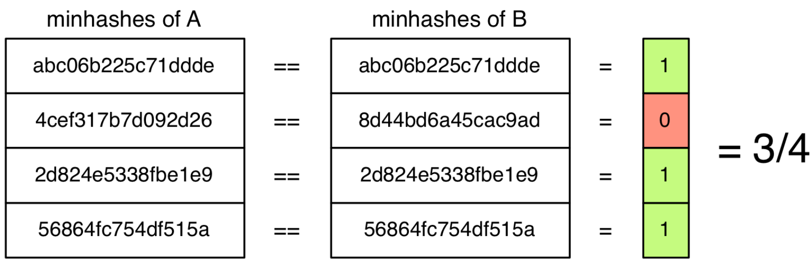


This needs long time to compute, because it needs to hash all shingles.

1. MinHash uses constant storage independent of the document length and producing a good estimate of our similarity measure.
2. This approach maps each document to a fixed-size set of hashes as a rough signature of this document.
3. This is accomplished by using a set of k randomizing hash functions.
4. For each randomizing hash function π_i, we pass the entire document's phrase hashes through to get a minimum hash denoted m_i.



1. The signature of the document is now the ordered list of these minimum hashes m_0 through m_{k-1}.
2. This method achieves an approximation to Jaccard similarity (the given probability).



Algorithm 5.3.1 Min Hash on set S

---

for $i = 1$ **to** N **do**
  if $(S(i) = 1)$ **then**
    for $j = 1$ to k **do**
      if $(h_j(i) < c_j)$ **then**
        $c_j \leftarrow h_j(i)$

- Now we have an extremely efficient method for estimating a Jaccard coefficient for a single pair of two documents.
- But we still have to estimate $O(N^2)$ coefficients where N is the number of web pages and still is intractable.
- A solution is locality sensitive hashing (LSH)

Spam pages

1. You have a page that will generate lots of revenue for you if people visit it.
2. Therefore, you would like to direct visitors to this page.
3. One way of doing this: get your page ranked highly in search results.

Spam technique: Keyword stuffing / Hidden text

1. Misleading meta-tags, excessive repetition
2. Hidden text with colors, style sheet tricks etc.
3. Used to be very effective, most search engines now catch these

Doorway page optimized for a single keyword, redirects to the real target page.

Lander page optimized for a single keyword or a misspelled domain name, designed to attract surfers who will then click on ads.

Spam technique: Duplication

1. Get good content from somewhere (steal it or produce it yourself)
2. Publish a large number of slight variations of it
3. For example, publish the answer to a tax question with the spelling variations of "tax deferred" on the previous slide

1. Create lots of links pointing to the page you want to promote
2. Put these links on pages with high (or at least non-zero) PageRank
    - Newly registered domains (domain flooding)
    - A set of pages that all point to each other to boost each other's PageRank
    - Pay somebody to put your link on their highly ranked page
    - Leave comments that include the link on blogs

1. Promoting a page in the search rankings is not necessarily spam.
2. It can also be a legitimate business – which is called SEO.
3. You can hire an SEO firm to get your page highly ranked.
4. There are many legitimate reasons for doing this.
   - For example, Google bombs like *Who is a failure?*
5. And there are many legitimate ways of achieving this:
   - Restructure your content in a way that makes it easy to index
   - Talk with influential bloggers and have them link to your site
   - Add more interesting and original content

1. Quality indicators
    - Links, statistically analyzed (PageRank etc)
    - Usage (users visiting a page)
    - No adult content (e.g., no pictures with flesh-tone)
    - Distribution and structure of text
2. Combine all of these indicators and use machine learning
3. Editorial intervention
    - Blacklists
    - Top queries audited
    - Complaints addressed
    - Suspect patterns detected

Web IR

Links The web is a hyperlinked document collection.

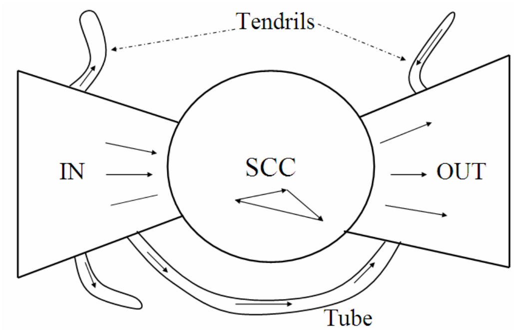
Queries Web queries are different, more varied and there are a lot of them. How many?

Users Users are different, more varied and there are a lot of them. How many?

Documents Documents are different, more varied and there are a lot of them. How many?

Context Context is more important on the web than in many other IR applications.

Ads and spam



1. Strongly connected component (SCC) in the center
2. Lots of pages that get linked to, but don't link (OUT)
3. Lots of pages that link to other pages, but don't get linked to (IN)
4. Tendrils, tubes, islands

1. Classic IR relevance (as measured by F) can also be used for web IR.
2. Equally important: Trust, duplicate elimination, readability, loads fast, no pop-ups
3. On the web, precision is more important than recall.
   - Precision at 1, precision at 10, precision on the first 2-3 pages
   - But there is a subset of queries where recall matters.

- Web search in most cases is interleaved with navigation (with following links).
- Different from most other IR collections
- Distributed content creation: no design, no coordination
- Unstructured (text, html), semistructured (html, xml), structured/relational (databases)
- Dynamically generated content

Size of the web

1. What is size? Number of web servers? Number of pages? Terabytes of data available?
2. Some servers are seldom connected (such as your laptop running a web server)
3. The dynamic web is infinite.

1. Random queries
2. Random searches
3. Random IP addresses
4. Random walks

Variant: Estimate relative sizes of indexes

1. There are significant differences between indexes of different search engines (max url depth, max count/host, anti-spam rules, priority rules etc.).
2. Different engines have different preferences.
3. Different engines index different things under the same URL (anchor text, frames, meta-keywords, size of prefix etc.).

- Generate a random URL
- Problem: Random URLs are hard to find (and sampling distribution should reflect user interest)
- Approach 1: Random walks / IP addresses : In theory: might give us a true estimate of the size of the web (as opposed to just relative sizes of index)
- Approach 2: Generate a random URL contained in a given engine: Suffices for accurate estimation of relative size

- Use vocabulary of the web for query generation
- Vocabulary can be generated from web crawl
- Use conjunctive queries w_1 AND w_2 (such as vocalists AND rsi)
- Get result set of one hundred URLs from the source engine
- Choose a random URL from the result set
- This sampling method induces a weight $W(p)$ for each page p.

1. Search for URL if the engine supports this or create a query that will find doc *d* with high probability.
   - Download doc, extract words
   - Use 8 low frequency word as AND query
   - Call this a strong query for *d*
   - Run query
   - Check if *d* is in result set
2. Problems
   - Near duplicates
   - Redirects
   - Engine time-outs

- Choose random searches extracted from a search engine log.
- Use only queries with small result sets.
- For each random query: compute ratio size(r_1)/size(r_2) of the two result sets
- Average over random searches

Summary

1. Many different approaches to web size estimation.
2. None is perfect.
3. The problem has gotten much harder.
4. There hasn't been a good study for a couple of years.
5. Great topic for a thesis!

Web crawler

1. Initialize queue with URLs of known seed pages
2. Repeat
   - Take URL from queue
   - Fetch and parse page
   - Extract URLs from page
   - Add URLs to queue
3. Fundamental assumption: The web is well linked.

What's wrong with the simple crawler

1. Scale: we need to distribute.
2. We can't index everything: we need to subselect. How?
3. Duplicates: need to integrate duplicate detection
4. Spam: need to integrate spam detection
5. Politeness: Web servers have policies (implicit/explicit) for regulating the rate at which a crawler can visit them. These policies must be respected.
6. Freshness: we need to recrawl periodically.
   - Because of the size of the web, we can do frequent recrawls only for a small subset.
   - Again, subselection problem or prioritization

What a crawler must do?

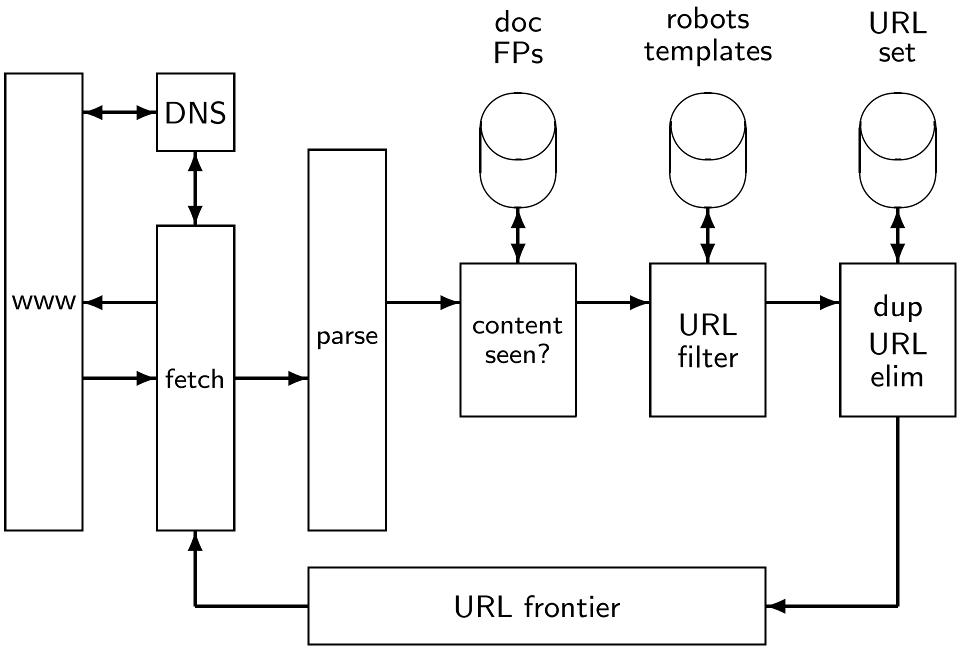
1. Be polite
   - Don't hit a site too often
   - Only crawl pages you are allowed to crawl: robots.txt
2. Be robust
   - Be immune to duplicates, very large pages, very large websites, dynamic pages etc

Robots.txt

1. Protocol for giving crawlers ("robots") limited access to a website, originally from 1994
2. Examples:
   - User-agent: *
     Disallow: /yoursite/temp/
   - User-agent: searchengine
     Disallow:
3. Important: cache the robots.txt file of each site we are crawling

A real crawler

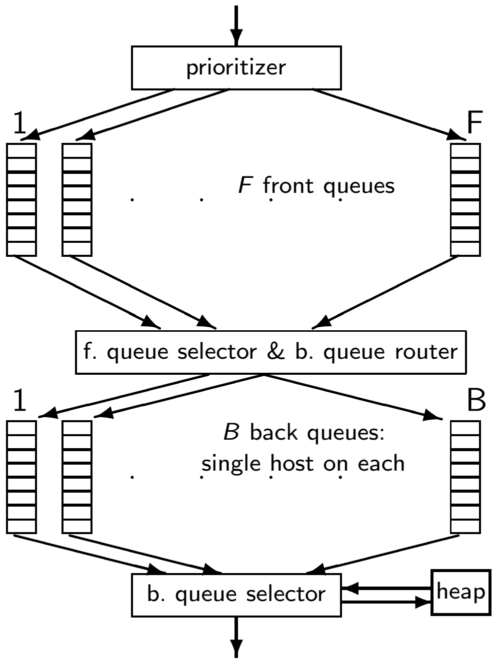
Basic crawl architecture



- The URL frontier is the data structure that holds and manages URLs we've seen, but that have not been crawled yet.
- Can include multiple pages from the same host
- Must avoid trying to fetch them all at the same time
- Must keep all crawling threads busy

URL frontier: Two main considerations

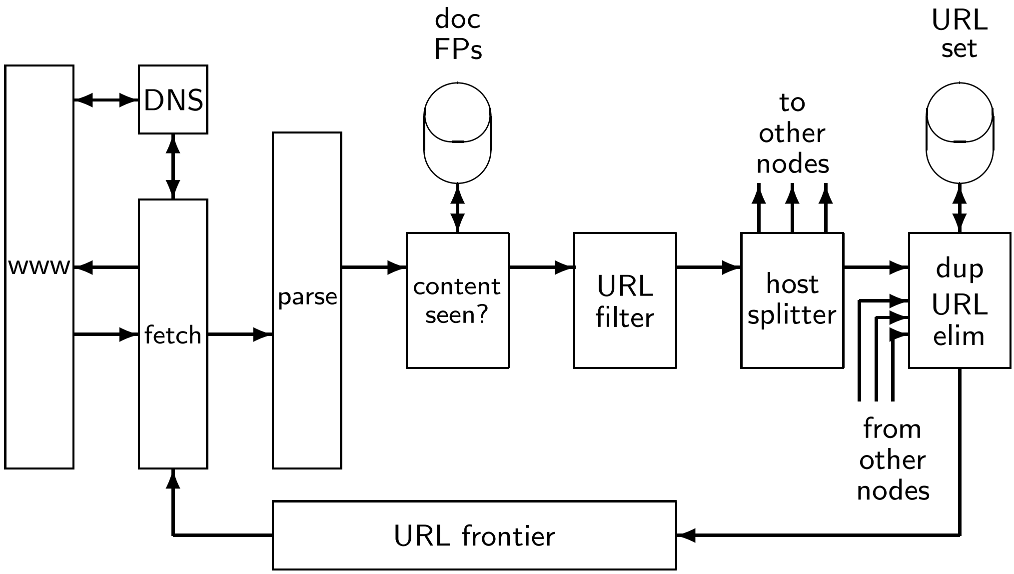
- Politeness: Don't hit a web server too frequently
  - E.g., insert a time gap between successive requests to the same server
- Freshness: Crawl some pages (e.g., news sites) more often than others
- Not an easy problem: simple priority queue fails.


prioritizer
1
F
F front queues
f. queue selector & b. queue router
1
B
B back queues:
single host on each
b. queue selector
heap

Distributing the crawler

1. Run multiple crawl threads, potentially at different nodes
2. Usually geographically distributed nodes
3. Partition hosts being crawled into nodes

Distributed crawl architecture



References

1. Chapters 19 and 20 of Introduction to Information Retrieval[1]

[1] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze (2008). *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press.

References

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze (2008). *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press.

Questions?