Information Retrieval and Text Mining

Information Retrieval on the web, search engines, ranking, PageRank, web crawling

(*this lecture is based on the support material provided together with Modern Information Retrieval textbook*)

Nuno Escudeiro (nfe@isep.ipp.pt)

Ricardo Almeida (ral@isep.ipp.pt)

Session outline

1. Introduction to Web Information Retrieval
2. Search engines
3. Search engine’s ranking
    - PageRank
4. Web crawling
    - Scheduling

Learning outcomes

At the end of this session, we will be able to:

- Describe the purposes, architecture and core tasks of web search engines
- Explain common search engine results' ranking functions
- Explain the rationale of PageRank and compute it
- Describe the tasks and the main challenges involved in web crawling

Session outline

1. **Introduction to Web Information Retrieval**
2. Search engines
3. Search engine’s ranking
    - PageRank
4. Web crawling
    - Scheduling

1. Introduction

Definition of IR

Information retrieval (IR) is the process of obtaining/finding relevant material/information to satisfy users' needs from large collections of documents normally stored on computers

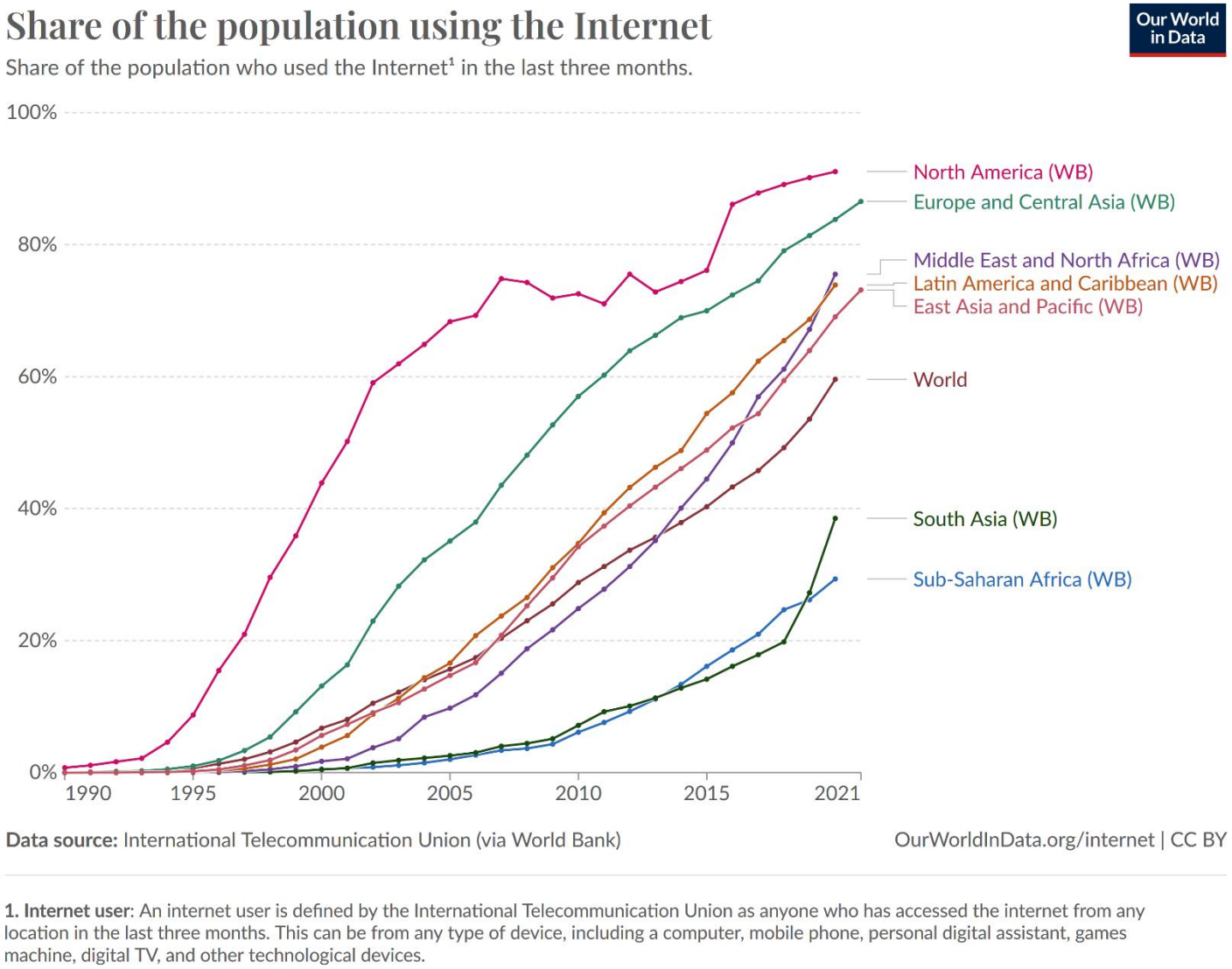
- IR deals with the representation, storage, organization of and access to information available in the form of:
  - Structured and semi structured records
  - Documents
  - Online catalogs
  - Multimedia objects
  - Hypertext documents/webpages.

Information Retrieval: core tasks

- IR normally involves:
  - **Indexing:** involves identifying and extracting important features or keywords from the documents, which are then stored in a searchable index. Creates an organized structure that facilitates efficient searching.
  - **Query Processing:** Users submit queries – requests for information or documents relevant to their information needs – usually as a set of keywords
  - **Ranking and Retrieval:** Retrieved documents are ranked based on their relevance to the user query
  - **Presentation:** Finally, the retrieved documents are presented to the user in a way that facilitates understanding and navigation, sorted by decreasing order of relevance.

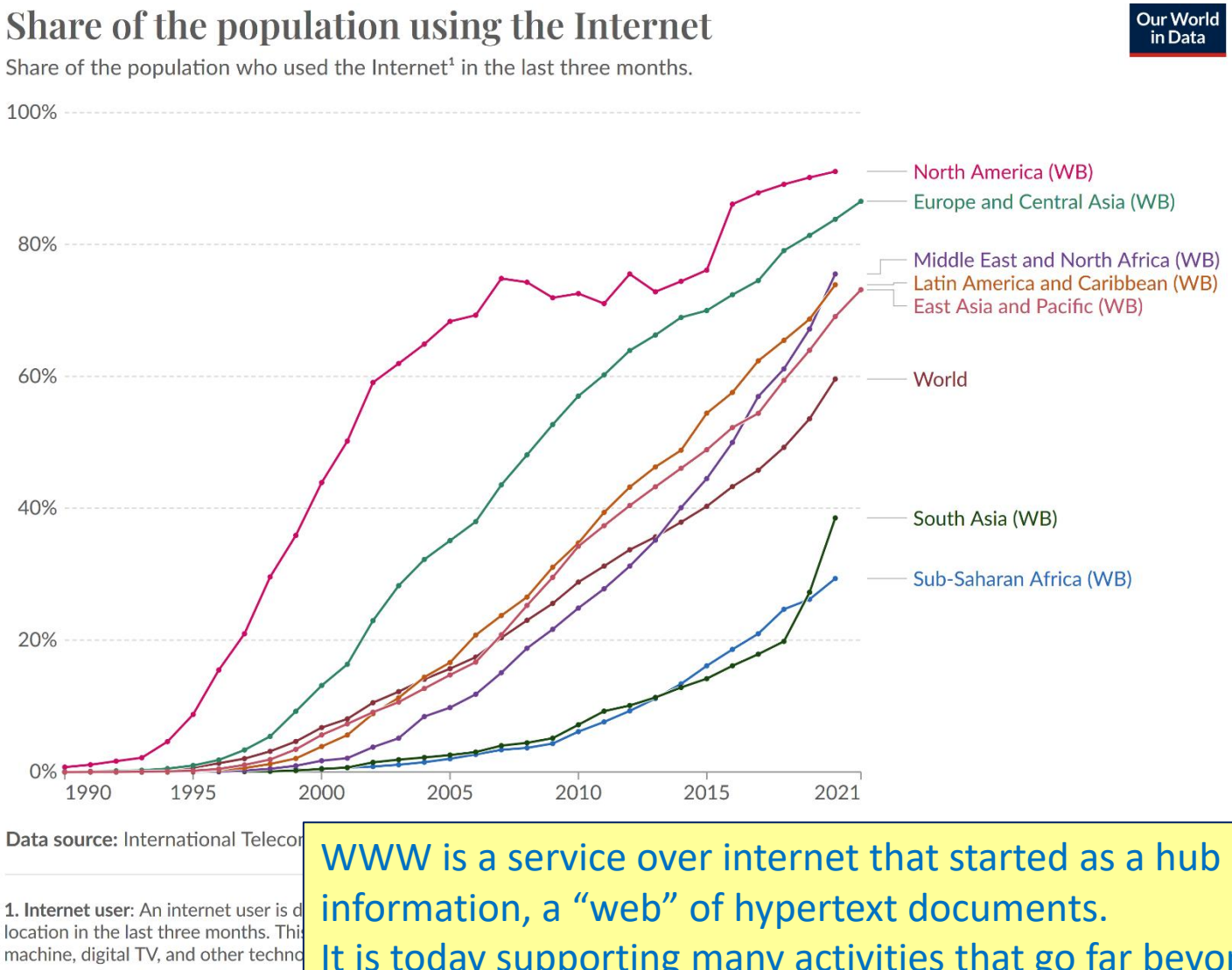
Information Retrieval on the Web

- The WEB boom characterized by the exponential growth on the volume of data, information and activities
- Web search engines are one of the most used tools on the web
- On the web, IR involves searching and retrieving web pages or online content based on user queries



Information Retrieval on the Web

- The WEB boom characterized by the exponential growth on the volume of data, information and activities
- Web search engines are one of the most used tools on the web
- On the web, IR involves searching and retrieving web pages or online content based on user queries



WWW is a service over internet that started as a hub of information, a “web” of hypertext documents.
It is today supporting many activities that go far beyond providing information: e-commerce, entertainment, social media, influencing.

Components of Web IR

- Crawling and Indexing
- Query processing and Retrieval of search results
- Ranking of search results
- Presenting search results.

Components of Web IR

Web Crawling

- Web Crawling also known as web scrapping, web spidering, web robot or bot is the process of systematically browsing WWW to discover and index web pages

- Role of Web Crawlers:
  - Discovering new web pages: Crawlers traverse the web by following links from one page to another, discovering new pages in the process
  - Building search engine indexes: Crawlers collect and index the content of web pages, making them searchable by search engines
  - Monitoring changes: Crawlers revisit previously crawled pages to detect changes and update the search engine index accordingly.

Components of Web IR

Web Crawling Components

- **Crawler**: responsible for fetching web pages, parsing their content, and extracting relevant information. It runs on a local system and send requests to remote web servers
- **Scheduler**: manages the scheduling of URLs to be crawled, ensuring efficient utilization of resources
- **Frontier**: queue of URLs to be crawled, prioritized based on factors like freshness and importance/quality

Web Crawling Process

- **Discovering new web pages**: crawlers traverse the web by following links from one page to another, discovering new pages in the process (depth first, breath first)
- **Building search engine indexes**: crawlers collect and index the content of web pages, making them searchable by search engines
- **Monitoring changes**: crawlers revisit previously crawled pages to detect changes and update the search engine index accordingly.

Components of Web IR

Web Crawling

- **Seed URLs**
  - The crawling process typically begins with a set of seed URLs, which are manually provided or generated based on known web sources
- **URL Extraction and Parsing**
  - Crawlers extract links from web pages by parsing HTML content
  - And apply techniques for filtering and normalizing URLs to avoid duplicates and ensure consistency
- **Crawling Policies**
  - Politeness: crawlers adhere to politeness policies to avoid overloading web servers and causing disruption. e.g: https://www.cloudflare.com/learning/bots/what-is-robots-txt/
  - Recrawl frequency: determining how often to revisit crawled pages based on factors like page importance and update frequency
  - Depth vs. breadth-first crawling: strategies for prioritizing crawling based on depth (depth-first) or breadth (breadth-first) of the web graph.

Components of Web IR

Indexing techniques

- Inverted indexes
- TDF-IDF
- Compression techniques.

Components of Web IR

Query Processing

- Retrieving relevant documents based on user queries
- Ranking algorithms and relevance scoring for ranking search results

Results Presentation

- Displaying search results to users in a user-friendly format
- Importance of snippets, titles, and metadata in search result presentation.

Session outline

1. Introduction to Web Information Retrieval
2. Search engines
3. Search engine's ranking
    - PageRank
4. Web crawling
    - Scheduling

2. Search Engines

Definition

- A search engine is a software system designed to search for information on the Web, retrieving relevant web pages based on user queries
- It acts as a gateway to the web, helping users discover, access, and navigate vast amounts of online content.

Role

- **Facilitating information retrieval**: Search engines enable users to find information quickly and efficiently, saving time and effort
- **Indexing the web**: Search engines crawl and index web pages, creating a searchable database of online content
- **Ranking search results**: Search engines use algorithms to rank search results based on relevance, authority, and other factors
  - Page **Authority** is a score that predicts how well a specific page will rank on search engine result pages (SERP) (https://moz.com/learn/seo/page-authority)
  - **Relevance** refers to the degree to which a search result meets the information needs of a user based on their query.

Relevance of ranking search results

- **Content Matching**: Relevance involves analyzing the content of web pages to determine how closely it aligns with the user's search query
- **Contextual Understanding**: Search engines aim to understand the context and intent behind a user's query to deliver relevant results
- **User Engagement Signals**: Relevance can also be influenced by user engagement signals, such as click-through rate (CTR), bounce rate, and dwell time
- **Authority and Trustworthiness**: Relevance is closely tied to the authority and trustworthiness of a web page or website. Search engines prioritize content from authoritative sources that are trustworthy and credible in their respective fields. Pages with high-quality backlinks, reputable domain authority, and positive user reviews are typically considered more relevant
- **Freshness and Recency**: For certain types of queries, freshness and recency of content can impact relevance. Search engines may prioritize recent or updated content for queries related to news, events, or rapidly evolving topics, ensuring that users receive the most up-to-date information
- **Multimedia and Rich Content**: Relevance extends beyond textual content to include multimedia elements such as images, videos, and interactive features.

Relevance of ranking search results

User Engagement Signals

Click-through rate: percentage of users who click on a search result after seeing it

A high CTR suggests that the result appears relevant based on the title/snippet shown

Bounce rate: percentage of users who leave the page without interacting or visiting other pages

A high bounce rate may indicate that the page did not meet user expectations or wasn't useful

Dwell time: amount of time a user spends on a page before returning to the search results

Longer dwell time often suggests the content was **engaging and relevant** to the user's query

- These metrics give search engines indirect feedback about **document relevance** from **real-world usage**:

  High CTR + Long dwell time → Positive signal

  High bounce rate → Negative signal.

Purpose

WWW is a service over internet that started as a hub of information, a “web” of hypertext documents.

It is today supporting many more activities going far beyond providing information: e-commerce, entertainment, social media, influencing

- **Information Retrieval**: The primary purpose of web search engines is to facilitate information retrieval from the vast amount of content available on the internet
- **Navigation**: Web search engines act as navigational tools, helping users explore the internet by providing links to relevant resources based on their search queries
- **Discovery**: Web search engines aid in the discovery of new websites, content, and resources that users may not have been aware of previously. By indexing and organizing web content, search engines make it discoverable to users worldwide
- **Problem Solving**: Web search engines assist users in solving problems by providing access to information, instructions, tutorials, and resources relevant to their queries
- **Decision Making**: Web search engines empower users to make informed decisions by providing access to diverse perspectives, reviews, opinions, and information on various topics. Users can compare products, services, and opinions before making decisions
- **Connectivity**: Web search engines serve as connectors, linking users to relevant online resources, communities, and networks.

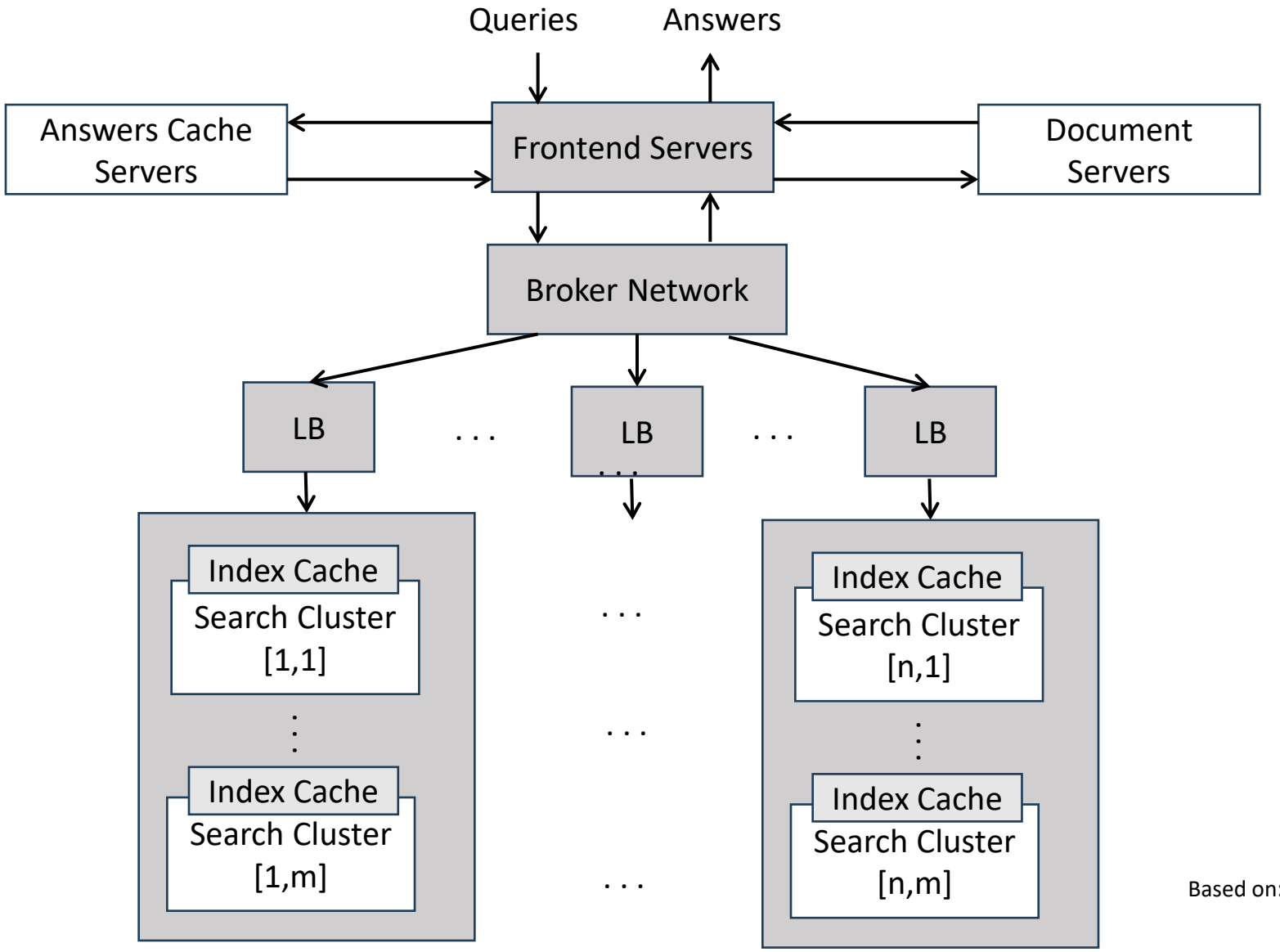
Examples

- https://www.google.com/
- https://www.yahoo.com/
- https://www.bing.com/
- https://www.ask.com/

Search Engine Architecture

- Current engines adopt a massively parallel cluster-based architecture
  - document partitioning is used
  - replicated to handle the overall query load
  - cluster replicas maintained in various geographical locations to decrease latency time (for nearby users)

Cluster-Based Architecture



Based on: Ricardo Baeza

Caching

- Search engines need to be fast
  - whenever possible, execute tasks in main memory
  - caching is highly recommended and extensively used
    - provides for shorter average response time
    - significantly reduces workload on back-end servers
    - decreases the overall amount of bandwidth utilized
- In the Web, caching can be done both at the client or the server side.

Caching

- Caching of answers
  - the most effective caching technique in search engines
  - query distribution follows a **power law**
    - small cache can answer a large percentage of queries
  - with a 30% hit-rate, capacity of search engine increases by almost 43%
- Still, in any time window a large fraction of queries will be unique
  - hence, those queries will not be in the cache
  - 50% in Baeza-Yates et al
  - this can be improved by also caching inverted lists at the search cluster level.

Multi-site Arquitecture

As the **document collection grows**

- capacity of query processors must grow as well
- the growth in size of single processors will unlikely match the growth of very large collections
  - even if numerous servers are used
  - main reasons are physical constraints such as size of single data center and power and cooling requirements.

Multi-site Arquitecture

Distributed resolution of queries using different query processors is a viable approach

- enables a more scalable solution
- but also imposes new challenges
- one such challenge is the routing of queries to appropriate query processors
  - to utilize more efficiently available resources and provide more precise results
  - factors affecting query routing include geographical proximity, query topic, or language of the query

Multi-site Arquitecture

Geographical proximity: reduce network latency by using resources close to the user posing the query

- Possible implementation is DNS redirection
  - according to IP address of client, the DNS service routes query to appropriate Web server
    - usually, the closest in terms of network distance
- As another example, DNS service can use the geographical location to determine where to route queries to
- There is a fluctuation in submitted queries from a particular geographic region during a day
  - possible to offload a server from a busy region by rerouting some queries to query servers in a less busy region.

Multi-site Arquitecture

- Baeza-Yates et al proposed a cost model for this kind of search engines
  - simple distributed architecture that has comparable cost to a centralized search architecture
  - architecture based on several sites that are logically connected through a star topology network
    - central site is the one with the highest load of local queries
  - **main idea**: answer local queries locally and forward to other sites only queries that need external pages in their answers
  - to increase percentage of local queries, use caching of results and replicate small set of popular documents in all sites
    - increase in number of local results from 5% to 30% or more.

Multi-site Arquitecture

- Cambazoglu et al show that
  - resources saved by answering queries locally can be used to execute a more complex ranking function
  - this can improve the results
- Cambazoglu et al also show that query processing can be improved by using linear programming to know when to re-route queries.

Session outline

1. Introduction to Web Information Retrieval
2. Search engines
3. **Search engine's ranking**
   - **PageRank**
4. Web crawling
   - Scheduling

3. Search Engine Ranking

Search Engine Ranking

Ranking is the hardest and most important function of a search engine

- **Key challenges**:
  - devise an adequate process of evaluating the ranking, in terms of relevance of results to the user
  - identification of quality content in the Web … the web is not edited … content is not verified/validated
  - avoiding, preventing, managing Web spam. Spammers are malicious users who try to trick search engines by artificially inflating signals used for ranking; a consequence of the economic incentives of the current advertising model adopted by search engines
  - defining the ranking function and computing it

Search Engine Ranking

Ranking is the hardest and most important function of a search engine

- **Key challenges**:
  - devise an adequate process of evaluating the ranking, in terms of relevance of results to the user
  - identification of quality content in the Web … the web is not edited … content is not verified/validated
  - avoiding, preventing, managing Web spam. Spammers are malicious users who try to trick search engines by artificially inflating signals used for ranking; a consequence of the economic incentives of the current advertising model adopted by search engines
  - defining the ranki

Quality indicators: domain name, textual contente, links, webpage layout (title, metadata, font size, etc.)

The highest traffic to the website, the more quality indicators will be available.

Ranking Signals

Distinct types of signals used for ranking: content, structure, or usage

Content signals

- related to the text itself
- can vary from simple word counts to a full IR score such as BM25
- can be provided by the layout, that is, the HTML source
  - simple format indicators (more weight given to titles/headings)
  - sophisticated indicators as the proximity of certain tags in the page

Ranking Signals

Structural signals

- intrinsic to the linked structure of the Web
- some of them are textual in nature, such as anchor text
- others pertain to the links themselves, such as in-links and out-links from a page
- link-based signals find broad usage beyond classic search engine ranking.

Ranking Signals

Web **usage signals** intrinsic to the linked structure of the Web

- main one is the **implicit feedback** provided by the user clicks (click-through), bounce rates and dwell time
- other usage signals include
  - information on the user's geographical context (IP address, language)
  - technological context (operating system, browser)
  - temporal context (query history by the use of cookies).

Link-based Ranking

- Number of hyperlinks that point to a page (in-links, authority) provides a measure of its popularity and quality
- Many common links in several pages are indicative of page relations with potential value for ranking purposes
- Examples of ranking techniques that exploit links are discussed next.

Early Algorithms

- Use incoming links for ranking Web pages
- … but … just counting links was not a very reliable measure of authoritativeness
  - easy to externally influence this count by creating new links to a page.

Early Algorithms

Yuwono and Lee proposed three early ranking algorithms

- **Boolean Spread**: begins with an initial result set with several pages *p* (e.g., based on **Boolean** or keyword matching).
  Then it **expands the result set** by including pages that are:
  - **linked from** page *p*, and
  - **linked to** page *p*.

  This models **local link structure** around initially relevant pages
- **Vector Spread**: similar in mechanism to Boolean Spread but applies in the **vector space model**. Instead of Boolean retrieval, it uses TF-IDF or cosine similarity to define the initial set of pages *p*.
  Then it **expands based on link neighborhood**, again using:
  - pages that point to or are pointed by the result pages *p*
- **Most-Cited**: a page's rank is influenced by how often **other pages (that cite it) contain query terms**.
  So if a page *p* is pointed to by many query-relevant pages, it gets a higher score.
  This is conceptually related to **citation count-based metrics** or early link-based scoring (predecessor of algorithms like PageRank).
  - a page *p* is assigned a score based on the total **number of query words contained in other pages** that point to page *p.*

PageRank Algorithm

HITS

Before diving into PageRank that look at HITS, a well-known ranking algorithm that is considered a predecessor of PageRank.

HITS (Hyperlink-Induced Topic Search) assigns two scores to each web page:

- **Authority score**: how valuable the page is on a topic (linked *to* by many hubs).
- **Hub score**: how useful the page is as a directory (links *to* many authorities).
- Pages reinforce each other: good **hubs point to good authorities**, and good **authorities are pointed to by good hubs**.

- Authority and Hub scores are computed iteratively.

HITS example

Suppose we have 4 pages:

- Page A links to B and C
- Page B links to C
- Page C links to none
- Page D links to C

A → B → C
A → C
D → C

HITS example

Iterative process

1 – **Initialize** all scores

Authority = 1, Hub = 1 for all pages

2 – **Update authority** scores

A page's authority = sum of the hub scores of pages linking to it

Example: Authority(C) = Hub(A) + Hub(B) + Hub(D) = 1 + 1 + 1 = 3

3 – **Update hub** scores

A page's hub = sum of the authority scores of pages it links to

Example: Hub(A) = Authority(B) + Authority(C) = 1 + 3 = 4

4 – **Normalize** scores and repeat until convergence

HITS example

Iterative process

1 – **Initialize** all scores

Authority = 1, Hub = 1 for all pages

2 – **Update authority** scores

A page's authority = sum of the hub scores of pages linking to it

Example: Authority(C) = Hub(A) + Hub(B) + Hub(D) = 1 + 1 + 1 = 3

3 – **Update hub** scores

A page's hub = su

Example: Hub(A)

4 – **Normalize** scores and repeat until convergence

Normalization of Authority and Hub scores:

- **L2 Norm** (standard): normalize scores so the sum of squares equals 1
- **L1 Norm**: normalize so the sum of scores equals 1
- Maximum: normalize to the maximum score
- Other are possible, although the L2 and L1 are those commonly used.

Outcome

- C becomes a strong authority (linked to by A, B, D)
- A becomes a strong hub (links to B and C)

PageRank

What is PageRank?

- An algorithm developed by Larry Page & Sergey Brin (1996) for ranking web pages
- Measures a page's "importance" based on incoming links, not just content

Why is it important?

- A foundational idea behind Google's early search engine success
- Still influences modern search ranking algorithms.

PageRank

Base idea

- A link from page A to page B is a vote of confidence in B
- More links → More authority

Assumptions

- Not all links are equal — links from important pages matter more
- A random surfer model: if a person keeps clicking links randomly, important pages are visited more often.

PageRank

PageRank of page p_i: $PR(p_i)$

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

N = total number of pages
$M(p_i)$ = set of pages linking to p_i
$L(p_j)$ = number of out-links from p_j
d = damping factor (d = 0.85 by default).

Principles of PageRank

1. Recursive definition

A page is important if it is linked to by other important pages

2. Link sharing

Each page's rank is distributed evenly among its out-links

3. Damping factor (d)

To avoid loops, it simulates a random jump with probability (1 - d), usually d = 0.85

PageRank

PageRank pseudo-code

Initialize PR[p] = 1/N for all pages p

Repeat until convergence, i.e., loop until changes between iterations are very small:

```
for each page p:
    PR_new[p] = (1 - d)/N
    for each page q that links to p:
        PR_new[p] += d * PR[q] / L(q)
PR = PR_new
```

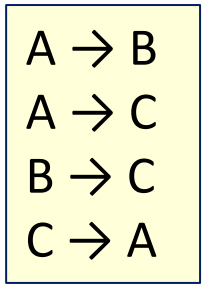
PageRank

Simple Web Graph:

Page A → Page B, Page C

Page B → Page C

Page C → Page A



Initial PR = 1/3 for each.

Apply formula with d = 0.85, iterate until convergence.

After convergence:

- Page C has the highest rank (most linked-to).
- Page A and B have lower ranks, depending on link structure.

PageRank

Summary

- PageRank captures importance from structure, not just content
- Based on random surfer behavior
- Still used as a feature in modern search engines, alongside content, context, and user behavior.

- PageRank matrix calculation, $PR_i = M^T PR_{i-1}$:

https://www.youtube.com/watch?v=3_1h13PJkUs

PageRank vs HITS

Feature	HITS	PageRank
Developed by	Jon Kleinberg (1998)	Larry Page & Sergey Brin (1996/1998)
Scores	**Two scores per page**: Hub & Authority	**One score per page**
Input	Applied to a **small subgraph** (from a query)	Global graph of the **entire web**
Focus	**Query-dependent** (starts from search results)	**Query-independent** (global rank)
Computation	**Iterative** link analysis **on query result graph**	**Iterative** analysis **on full link structure**
Sensitivity	**Sensitive** to spam and tightly connected clusters	More **robust**, more resistant to link spam

- **HITS inspired later developments**, including ideas around link-based authority.
- **PageRank** generalized the idea, focusing on **overall importance** rather than just topical relevance.
- In modern systems, elements of **both** are often combined or adapted.

Session outline

1. Introduction to Web Information Retrieval
2. Search engines
3. Search engine's ranking
   - PageRank
4. **Web crawling**
   - **Scheduling**

4. Web Crawling

Brief History

- The first known web crawler was created in 1993 by Matthew Gray, an undergraduate student at MIT
- The project of this Web crawler was called WWWW (World Wide Web Wanderer)
- It was used mostly for Web characterization studies
- In June 1994, Brian Pinkerton, a PhD student at the University of Washington created a WebCrawler index for searching with an index based on the contents of documents located on nearly 4000 servers
- Other search engines based on Web crawlers appeared: Lycos (1994), Excite (1995), Altavista (1995), and Hotbot (1996)
- Currently, all major search engines employ crawlers.

Applications of a Web Crawler

- create an index covering broad topics (general Web search)
- create an index covering specific topics (vertical Web search)
- archive content (Web archival)
- analyze Web sites for extracting aggregate statistics (Web characterization)
- keep copies or replicate Web sites (Web mirroring)
- web site analysis.

General Web Search

- Web search has driven web crawling development
- Types of Web search
  - **General Web search**: done by large search engines
  - **Vertical Web search**: the set of target pages is delimited by a topic, a country or a language
- Crawlers for general web search must balance coverage and quality
  - **Coverage**: must scan pages that can be used to answer many different queries
  - **Quality**: the pages should have high quality.

Vertical Web Search

Vertical Crawler: focus on a particular subset of the Web

- This subset may be defined geographically, linguistically, topically, etc.
- Examples of vertical crawlers
  - Shopbot: designed to download information from on-line shopping catalogs and provide an interface for comparing prices in a centralized way
  - News crawler: gathers news items from a set of pre-defined sources
  - Spambot: crawler aimed at harvesting e-mail addresses inserted on Web pages
  - Tumba: crawler (search engine) for the Portuguese web.

Vertical Web Search

Vertical search also includes **segmentation by a data format**

- In this case, the crawler is tuned to collect only objects of a specific type, as image, audio, or video objects
- Example
  - Feed crawler: checks for updates in RSS/RDF files in Web sites.

Topical Crawling

Focused crawlers: focus on a particular topic

- Provides a more efficient strategy to avoid collecting more pages than necessary
- A focused crawler receives as input the description of a topic, usually described by
  - a driving query
  - a set of example documents
- The crawler can operate in
  - batch mode, collecting pages about the topic periodically
  - on-demand, collecting pages driven by a user query

Web Characterization

Web characterization includes all attempts to derive statistical properties of Web pages

- difficult questions regarding Web characterization
  - what constitutes a representative sample of the Web?
  - what constitutes the Web of a country?
- crawled pages deeply affect the results of the characterization
- page-centered characterization are less affected than link-centered characterization efforts
- the **seed URLs** must be chosen carefully

Mirroring

Mirroring is the act of keeping a partial or complete copy of a Web site

- Crawlers used for mirroring are usually simpler
- Mirroring policy includes:
  - the refreshing period, typically daily or weekly
  - the time of the day to do the mirroring.

Web Archiving

Web archiving is a mirroring without discarding the outdated copies, that is, the whole history of each page is recorded

- The largest project of Web archiving is the Internet Archive:
  - http://www.archive.org/
  - Its main purpose is to preserve the state of the Web on each year.

Web Site Analysis

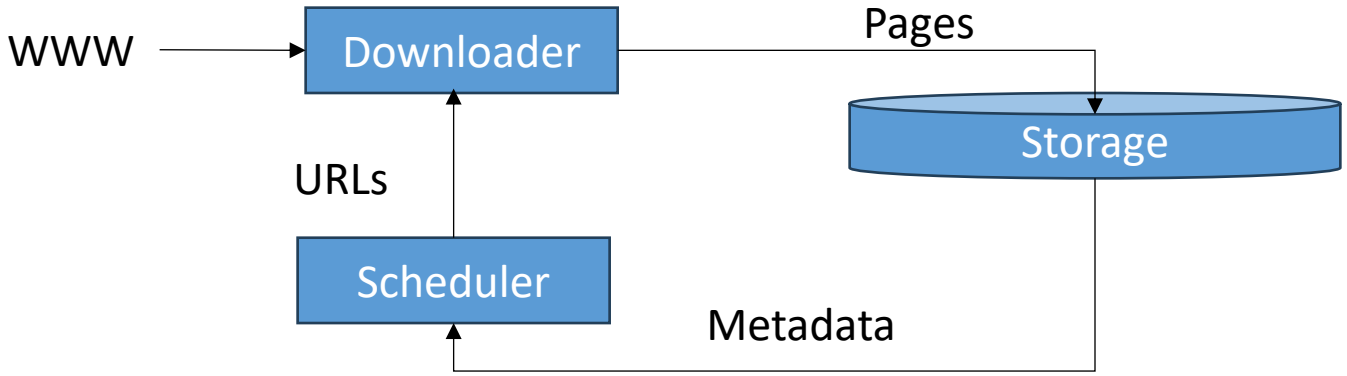
- A Web crawler can be used to analyze a Web site and even change it on the fly according to a predefined criteria
- Examples of automatic site analysis:
  - **Link validation**: scanning the pages of a site for broken links
  - **Code validation**: ensures that all pages of a site are well-formed
  - **Web directories analysis**: looking for sites that are no longer available
- A site analysis tool can also be used to **find vulnerabilities** in Web sites
  - Including to find older and unpatched versions of popular scripts
- In large text repositories, Web crawlers can be used to **automate many tasks**, including:
  - Categorization, to ensuring that all pages in a set conform to a standard
  - Detection of images with unknown copyright status
  - Detection of orphan (unlinked) pages.

Taxonomy of Crawlers

- The crawlers assign different importance to issues such as freshness, quality, and volume
- The crawlers can be classified according to three axes:
  - **Freshness**
  - **Quality**
  - **Volume**

Arquitecture of the Crawlers

- The crawler is composed of three main modules: downloader, storage, and scheduler
  - **Scheduler**: maintains a queue of URLs to visit (frontier)
  - **Downloader**: downloads the pages
  - **Storage**: makes the indexing of the pages, and provides the scheduler with metadata on the pages retrieved



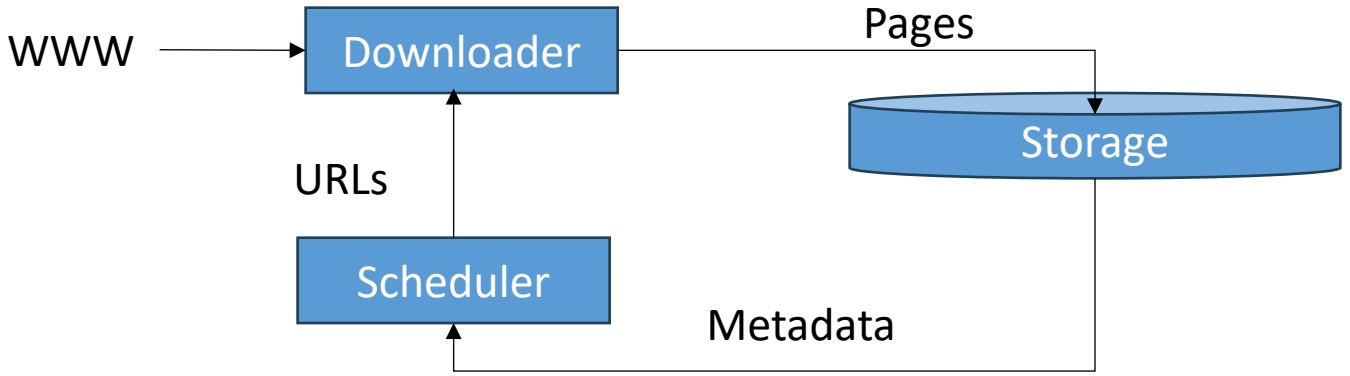
Arquitecture of the Crawlers

- The crawler is composed of thre
  storage, and scheduler
  - **Scheduler**: maintains a queue of U
  - **Downloader**: downloads the page
  - **Storage**: makes the indexing of th
    metadata on the pages retrieved

Scheduling can be further divided into two parts:
- Long-term scheduling: decide which pages to visit next
- Short-term scheduling: re-arrange pages to fulfill politeness.

Enforcement of the politeness policy requires maintaining several queues, one for each site, and a list of pages to download in each quele.

Storage can also be further subdivided into three parts:
- (Rich) text
- Metadata and
- Links



Practical Issues

- The implementation of a crawler involves many practical issues
- Most of them are due to the need to **interact with many different systems**
- Example: How to download maintaining the traffic produced as uniform as possible?
  - The pages are from multiple sources
  - DNS and Web server response times are highly variable
  - Web server up-time cannot be taken for granted

- Other practical issues are related with: **types of Web pages**, **URL canonization**, **parsing**, **wrong implementation of HTML standards**, **broken links** and **duplicates**

Taxonomy of Web Pages

- A challenging aspect is related with the taxonomy of Web pages
- Web pages can be classified in public or private, and static or dynamic
  - There are in practice infinitely many **dynamic pages**
  - We cannot expect a crawler to download all of them
  - Most crawlers choose a maximum depth of dynamic links to follow.

Wrong HTML

- Most Web browsers are very tolerant with **HTML wrongly coded**
  - This has led to very poor quality in the HTML coding
  - The parser module of the crawler must allow for mistakes in the HTML coding
- In many cases it is difficult to tell if a **link is broken** or not
  - Some servers use a custom-built error page to show that a page does not exist, without send a response header signaling the error condition
  - Bar-Yosef et al refer to these error pages as soft-404, and observe that 29% of dead links point to them
  - Some Web crawlers test Web sites by sending a URL that (probably) does not exist

Duplicates

- The prevalence of mirrored content on the web is high
- Types of duplicates
  - intentional duplicates: mirroring of other pages
  - unintentional duplicates: the result of the way that many Web sites are built
- The worst of unintentional are caused by identifiers embedded in the URLs to track user's behavior
  - e.g.: /dir/page.html&jsessid=09A89732
  - a Web crawler must be aware of session-ids and try to keep a consistent session-id across requests

Granularity of Information

- **Blogs**, **Web forums**, and **mailing list archives**: large repositories of information comprised of many small postings
- Useful source of information when the topic is not covered somewhere else
- However, sometimes individual postings are not valuable
- A Web crawler might index only the pages that aggregate information

Parallel and Distributed Crawling

- To achieve better scalability and be more tolerant to failures, Web crawling should be done in parallel and distributed fashion
- In this case, the most important issue is to avoid downloading the same page more than once and/or overloading Web servers
- The coordination among processes is done by exchanging URLs
- The goal of the crawler designer is to minimize the communication overhead
  - Ideally, every page should be downloaded by a single process

Parallel and Distributed Crawling

- A fully distributed crawling system requires a **policy for assigning the new URLs discovered**
- The decision of which process should download a given URL is done by an **assignment function**
- Boldi et al state that an effective assignment function must have three main properties:
  - **Balancing property**: each crawling process should get approximately the same number of hosts
  - **Contra-variance property**: if the number of crawling processes grows, the number of hosts assigned to each process must shrink
  - The assignment must be able to **add and remove crawling processes dynamically**

Scheduling Algorithms

Scheduling Algorithms

- A Web crawler needs to balance various objectives that contradict each other
  - It must download new pages and seek fresh copies of downloaded pages
  - It must use network bandwidth efficiently avoiding to download bad pages
  - However, the crawler cannot know which pages are good, without first downloading them
- To further complicate matters, there is a huge number of pages being added, changed and removed every day on the Web.

Scheduling Algorithms

- The simplest crawling scheduling is traversing Web sites in a **breadth-first** fashion
  - This algorithm increases the Web site coverage
  - And is good to the politeness policy by not requesting many pages from a site in a row
- However, can be useful to consider the crawler's behavior as a combination of a series of policies
- To illustrate, a crawling algorithm can be viewed as composed of three distinct policies
  - **selection policy**: to visit the best quality pages, first
  - **re-visit policy**: to update the index when pages change
  - **politeness policy**: to avoid overloading Web sites

Selection Policy

- Even large search engines cover only a portion of the publicly available content
- It is highly desirable that such downloaded fraction contains the most authoritative pages
- A crawler must carefully choose, at each step, which pages to visit next
- The selection of which pages to crawl can be divided into **two types of restrictions**
  - **Off-line limits** that are set beforehand
  - **On-line selection** that is computed as the crawl goes by

Off-line Limits

- Due to storage limitations, in practice, it is frequently necessary to establish beforehand limits for the crawling process
- The off-line limits used more frequently by Web crawlers are the following
  - A maximum number of hosts to be crawled
  - A maximum depth (a maximum number of links to be traversed starting from any home page)
  - A maximum overall number of pages in the collection
  - Maximum number of pages or bytes downloaded from each server
  - A list of accepted mime-types for downloading (e.g.: text/html and text/plain).

On-line Selection

- A crawler requires a **metric of importance for prioritizing Web pages**
- The importance of a page may be a function of
  - its intrinsic quality
  - its popularity in terms of links or visits
  - its URL (in the case of vertical search engines restricted to a top-level domain or to a fixed Web site)
- Another difficulty: the crawler must work with partial information, as the complete set of Web pages is not known during crawling

Re-visit Policy

- The Web has a very dynamic nature
- By the time a Web crawler has finished its crawl, many events might have happened
- We characterize these events as creations, updates and deletions:
  - **Creations**: when a new page is created (and becomes accessible by a link) it can be crawled as determined by the visit politic
  - **Updates**: an update can be either minor (occurs at the paragraph or sentence level), or major (all references to its content are not valid anymore)
  - **Deletions**. Undetected deletions of pages are more damaging for a search engine's reputation than updates.

Modeling of Page Events

- From the search engine's point of view, there is a cost associated with not detecting an event
- The cost functions used more frequently are **freshness** and **age**
- **Freshness** is a binary measure that indicates whether the local copy is up-to-date or not
- The freshness of a page p in the repository at time t is defined as:

$$F_p(t) = \begin{cases} 1, if\ p\ is\ equal\ to\ the\ local\ copy\ at\ time\ t \\ 0, otherwise \end{cases}$$

Modeling of Page Events

- **Age** is a measure that indicates how outdated the local copy is
- The age of a page p in the repository, at time t is defined as:

$$A_p(t) = \begin{cases} 0, if\ p\ has\ not\ been\ modified\ at\ time\ t, since\ last\ update \\ t - lu(p), otherwise \end{cases}$$

where lu(p) is the last update time for the page p

Evolution of freshness and age, two types of events may occur:

- **Event modify**, modification of a Web page in the server
- **Event sync**, downloading of the modified page by the crawler

Strategies

- The **objectives of the crawler** can be:
  - to keep the average freshness of pages in the collection as high as possible, or
  - to keep the average age of pages in the collection as low as possible

- These are not equivalent objectives:
  - In the first case, the crawler is just concerned with how many pages are out-dated
  - In the second case, the crawler is concerned with how old are the local copies of the pages.

Politness Policy

- The use of Web robots, while useful for a number of tasks, comes with a price for the general community
  - Web crawlers require considerable bandwidth
  - They can create server overload, specially if the frequency of access to a given server is high, and/or if the robot is poorly written
- Privacy is also an issue with Web crawlers
  - They may, for instance, access parts of a Web site that were not meant to be public
  - If the crawler keeps a cache of pages, copyright issues which are currently not enforced may arise

Politness Policy

- A set of guidelines is also important for the continued operation of a Web crawler
- **A crawler that is impolite with a Web site may be banned by the hosting provider**
- The **three basic rules for Web crawler operation** are:
  - A Web crawler must **identify itself as such**, and must not pretend to be a regular Web user
  - A Web crawler must **obey the robots exclusion protocol** (robots.txt)
  - A Web crawler must **keep a low bandwidth usage in each Web site**

Politness Policy

- Google (Googlebot)

  http://www.google.com/webmasters/bot.html

- Yahoo! Search (Slurp!)

  http://help.yahoo.com/help/us/ysearch/slurp/

Robot Identification

- Sometimes, the navigational pattern of a Web crawler may be detected by a Web server
- However, this detection is more effective if the Web crawler identifies itself as such in the first place
- The HTTP protocol includes a user-agent field that can be used to identify who is issuing a request
- The user-agent field of a Web crawler should include an address to a Web page containing information on the crawler, as well as contact information
- When this information is not present, Web site administrators might send complaints to the listed owner of the entire originating network segment.

Robot Exclusion Protocol

- The robot exclusion protocol involves three types of exclusion:
  - server-wide,
  - page-wise exclusions, and
  - cache exclusions.

Robot Exclusion Protocol

- **Server-wide exclusion** instructs the crawler about directories that should not be crawled
- This is done via a single robots.txt file that is located in the root directory of a Web site.

Robot Exclusion Protocol

- **Page-wise exclusion** is done by the inclusion of meta-tags in the pages themselves
- Meta-tags are part of the standard HTML syntax and allow a page author to associate pairs of the form key=value to Web pages.

Robot Exclusion Protocol

- **Cache exclusion** is used by publishers that sell access to their information
- While they allow Web crawlers to index the pages, they instruct search engines not to show the user a local cached copy of the page
- Even with all the precautions, a crawler might access pages that were not meant to be public
- Because of this, it is important to have a fast way of removing a document from the local collection.

Controlling Bandwidth Usage

- The bandwidth available for a crawler is usually much higher than the bandwidth of the Web sites it visits
- Using multiple threads, a Web crawler might easily overload a Web server, specially a smaller one
- To avoid this, it is customary
  - to open only one connection to a given Web server at a time
  - to take a delay between two consecutive accesses
- Recently, several Web crawlers allow Web site operators to decide which is the delay that should be used when indexing their site
- This is done by the robots exclusion protocol, including in the robots.txt a line specifying a crawl-delay.

Combining Policies

- The behavior of the crawler can be separated into two parts
  - a short-term scheduling, dealing with the politeness policy
  - a long-term scheduling, dealing with selection and freshness
- A natural combination of these policies is to consider the profit obtained from downloading a single Web page
- Suppose that a local page has an estimated quality q and that its probability of being up-to-date is p
  - Then we can consider that the value of the page in the index is $q \times p$

Combining Policies

- If we download the page now, its probability of being up-to-date becomes 1, so its value becomes q
  - Then, the expected profit of downloading a page is $q \times (1 - p)$ which will be 0 for pages just downloaded
- A natural policy then is to sort pages by expected profit
- Other types of decay can be used to account for the pages that are not “fresh” in the repository
- Notice that to have more pages we need to crawl new pages instead of refreshing pages, but the only way to find new pages is refreshing pages that have a new link in them

References

- https://users.dcc.uchile.cl/~rbaeza/mir2ed/slides.php.html
- Query expansion techniques for information retrieval: A survey Hiteshwar Kumar Azad*, Akshay Deepak

(https://www.sciencedirect.com/science/article/pii/S0306457318305466?via%3Dihub#bib0051)