Introduction to the Web

Maria Maistro
mm@di.ku.dk

Web Science Lecture
8 February 2021

UNIVERSITY OF COPENHAGEN

Credit for many of the slides: Christina Lioma and Isabelle Augenstein

GENERAL COURSE INFORMATION

Teaching Team

Course Responsible:

- Maria Maistro, mm@di.ku.dk

Lecturers:

- Maria Maistro
- Sagnik Ray Choudhury, src@di.ku.dk

Labs:

- Ziheng Liu (Nico), swl480@alumni.ku.dk

Course Organization

- Prerecorded lectures released on Friday at 10:00;
- Every Monday from 11:00 to 11:45 there will be an online Q&A session;
- Every Tuesday from 13:00 to 15:00 there will be an online TA session.

Course Info: Absalon

Absalon:

- Lecture plan, projects, readings, slides, latest news and other **important** information;
- Keep an eye on the course homepage **throughout** the block for information updates;
- **Familiarise** yourselves with Absalon;
- Last minute changes (e.g. class cancellation, change in Covid guidelines, …).

Also – course description:
https://kurser.ku.dk/course/ndak14004u/2020-2021

Course Info: Slack Channel

- Join the slack channel for Web Science 2021:

https://join.slack.com/t/webscience2021/shared_invite/zt-lv0n1f43-UvbkzOA_M9RwurB30eswIw

- Remember: Absalon is the course **official** channel, all official announcements will be posted there.
- Please do not upload big files on slack (free license).

Course Info: Resources

Readings:

- On Absalon course page (no single textbook);
- Provide important context to **supplement** lectures, but they do not replace lectures.

Lectures:

- Slides supplement the pre-recorded lectures, but they do not replace them;
- Pointers to more readings and sources;
- Q&A sessions for questions and discussions.

Labs:

- Help with the projects, but not solve them for you;
- Answer questions about the project or lectures.

Things to Know

Attendance:

- Your responsibility to attend; if not, no *formal* way of catching up.

Plagiarism:

- Automatic fail on project;
- Referral to head of students.

Prerequisites:

- Programming;
- Machine Learning.

To Pass the Course

1. Continuous project (throughout the course):
   - Individual;
   - Written report and code has to be handed in.

2. Oral defence:
   - Individual;
   - 10-minute project presentation (online), followed by 10 minutes of questions;
   - In exam week.

Final grade based on overall assessment (not average).

Re-exam

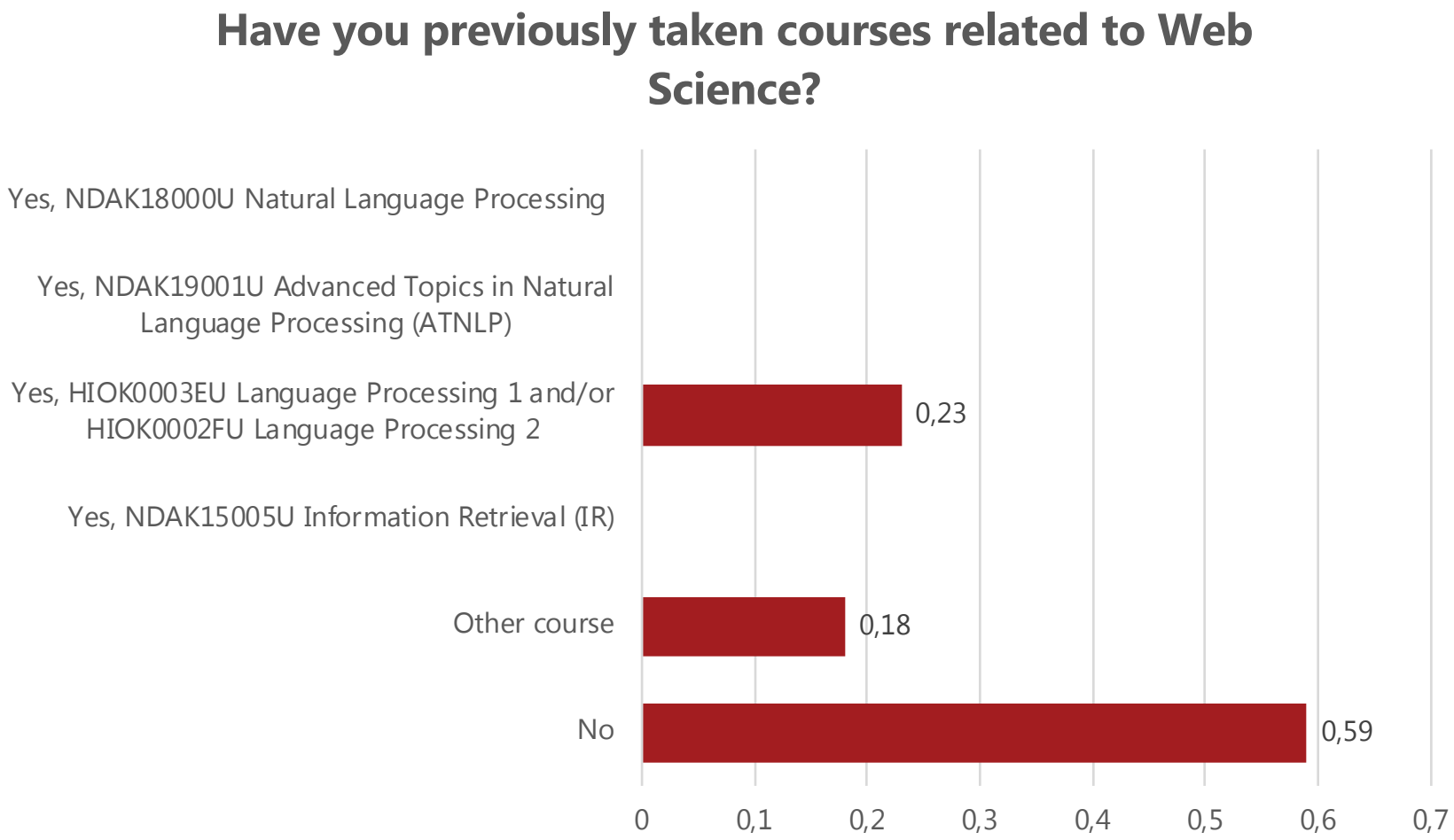
1. New project;
2. Oral exam on the full course syllabus without preparation.

Both 1 AND 2 are compulsory:

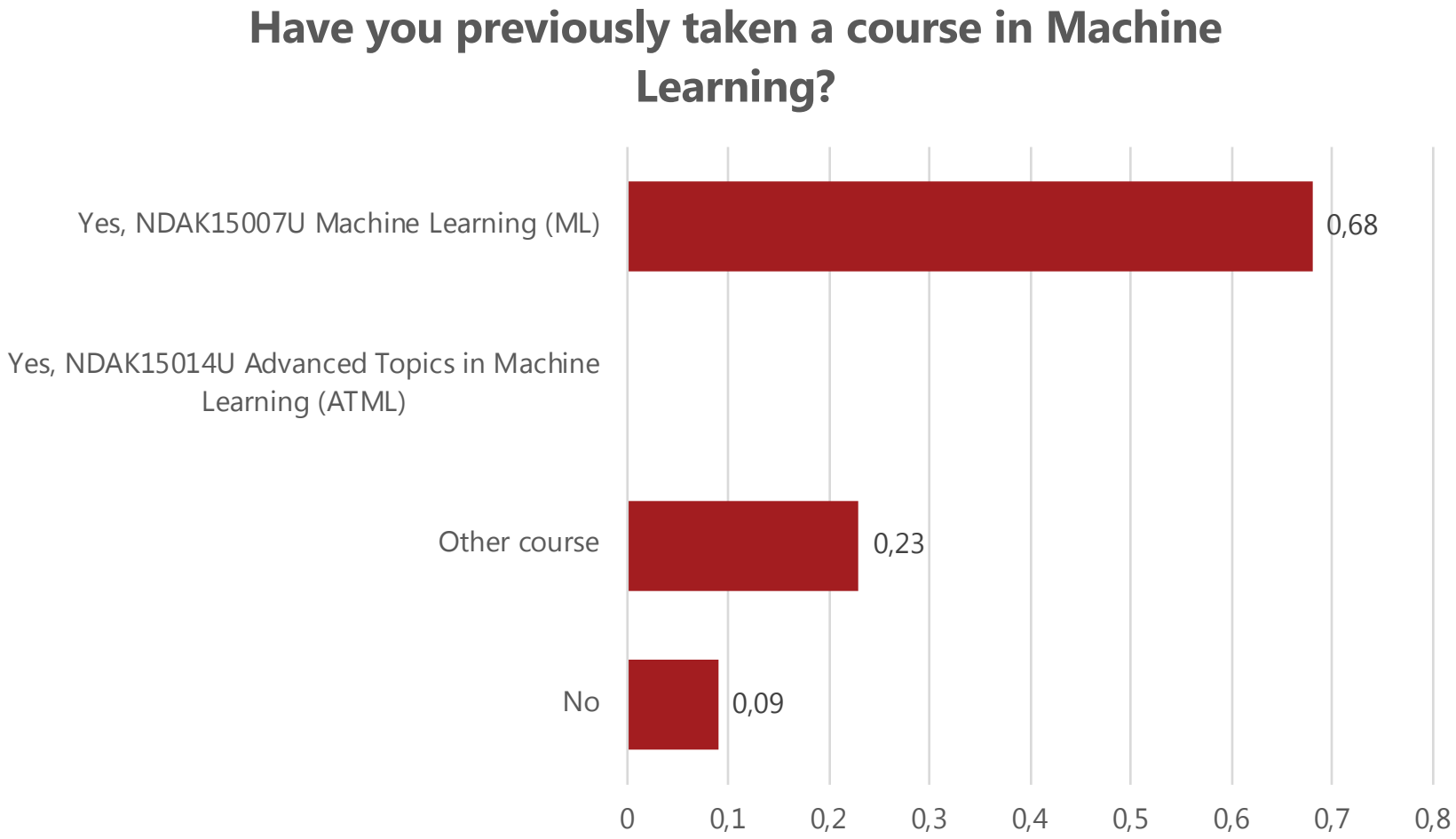
- If you do not submit the new projects, you cannot take the oral exam → automatic fail
- If you do not show up at the oral → automatic fail

Final grade based on overall assessment (not average).

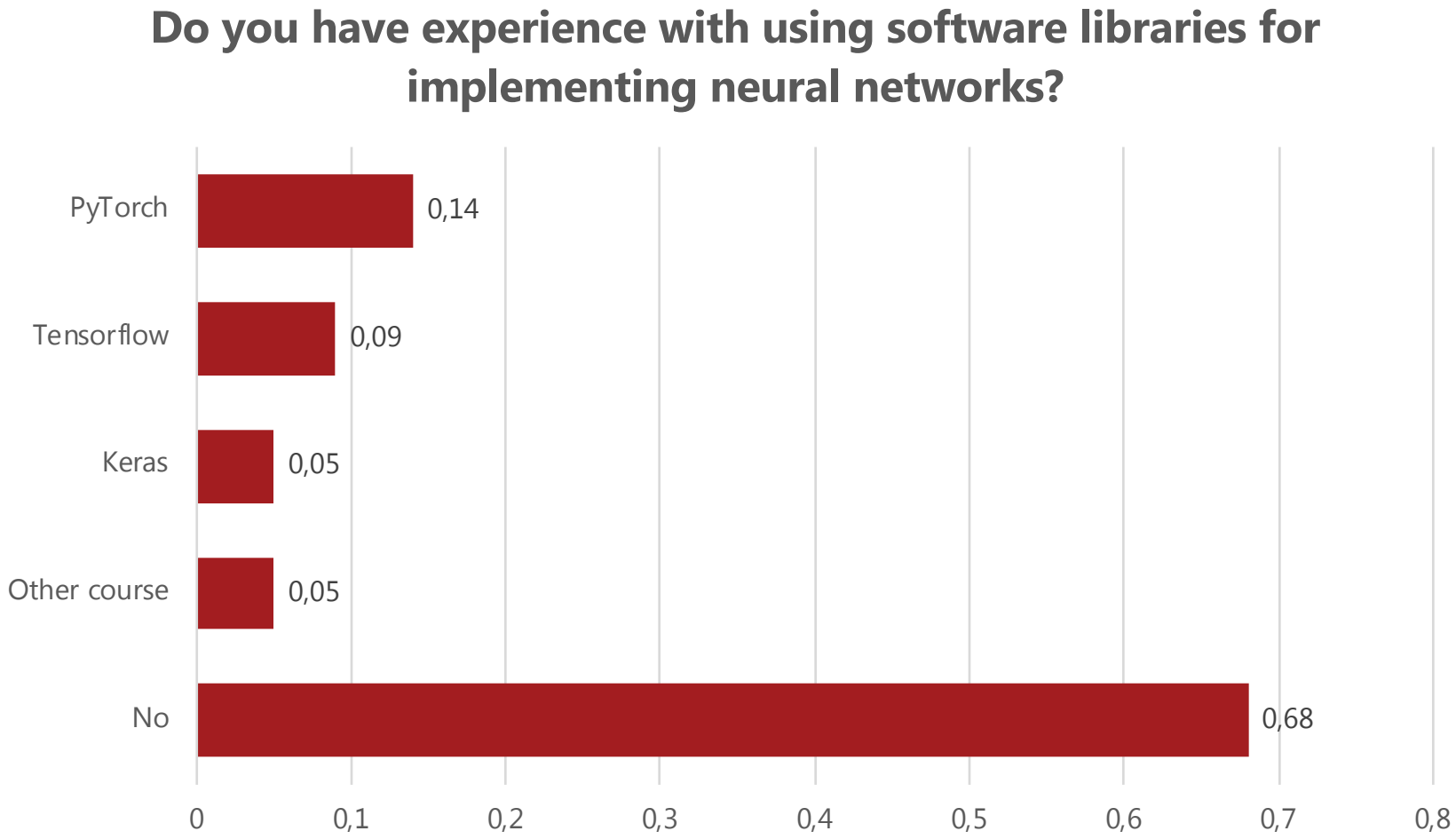
About You – Results of Quiz



About You – Results of Quiz



About You – Results of Quiz



About You – Results of Quiz

What do you want to get out of Web Science?

For example, are you mainly taking this course for academic credits, because you think it will improve your job prospects, or for some other reason?

- Want to learn about WWW (4)
- Academic and job related reasons (4)
- Want to learn about Web Crawling (3)
- Want to learn about Data Mining/Web Mining (2)
- Want to learn about NLP (2)
- Recommended by a peer (2)
- Master thesis (1)
- Want to learn about recommender systems (1)

Today's Lecture

- Introduction to Web Science and the Web;
- Main challenges of Web data processing;
- Web crawling;
- Web Crawling in Practice.

INTRODUCTION TO WEB SCIENCE AND THE WEB

What is Web Science?

Web Science: **non-trivial** processing and **application** of implicit, previously unknown, potentially useful information from **Web data**

Web data examples: Web pages, blogs, tweets, Amazon clicks, YouTube comments, dr.dk updates, sourceforge downloads,...

Application examples: opinion mining, trend detection, recommendation, Web search, crowdsourcing, web analytics...

Borrows methods from: machine learning, information retrieval, natural language processing, data mining, databases, statistics...

Spans from back-end (algorithms) to front-end (visualisation).

Different Scientific Areas

Web Science borrows methods from: machine learning, information retrieval, natural language processing, data mining, statistics…

Different scientific cultures

- *Machine Learning:* focus on "complex" methods, "small" data;
- *Information Retrieval:* large-scale (non main-memory) data;
- *Natural Language Processing:* focus on fine-grained processing of text and/or linguistictasks;
- *Data Mining:* focus on discovering patterns in data;
- *Visualisation:* focus on user-intuitive data overviewing;
- *Data cleansing:* focus on detecting bogus data, e.g. age=150.

Question

What is the difference between the WWW and Internet?

A. There is no difference, both the Internet and WWW term refers to the same network (infrastructure with a service);
B. Internet is a network of computers (infrastructure) while the WWW is an information sharing model (service);
C. The WWW is a network of computers (infrastructure) while internet is an information sharing model (service);
D. The WWW is a network of computers (infrastructure) and internet part of the WWW (hardware).

Web and Internet

Web: not the same as the Internet;

Internet: massive network of computers that can communicate (transfer data between them) in various languages called *protocols;*

World Wide Web (or Web): information-sharing model built on top of the Internet, which uses the HTTP protocol to transfer data;

Web: one of many ways to share information on the Internet. Other ways are email (SMTP protocol), instant messaging (SIMPLE), FTP,...

The Web is *part of* the Internet.

Web as a Graph (Directed)

- A graph G = (V, E) is defined by:
  - a set V of vertices (nodes);
  - a set E of edges (links) connecting pairs of nodes.
- The **Web page graph** (**directed**):
  - V is the set of pages;
  - E is the set of hyperlinks (**inlinks & outlinks**);
  - ~10-20 hyperlinks per Web page on average.
- **Power law** distribution of hyperlinks:
  - Very few pages have the most hyperlinks;
  - Vast majority of pages have very few hyperlinks.
- Many more graphs can be defined:
  - e.g. Twitter mention graph, co-citation graph.

Bow-Tie Structure of the Web

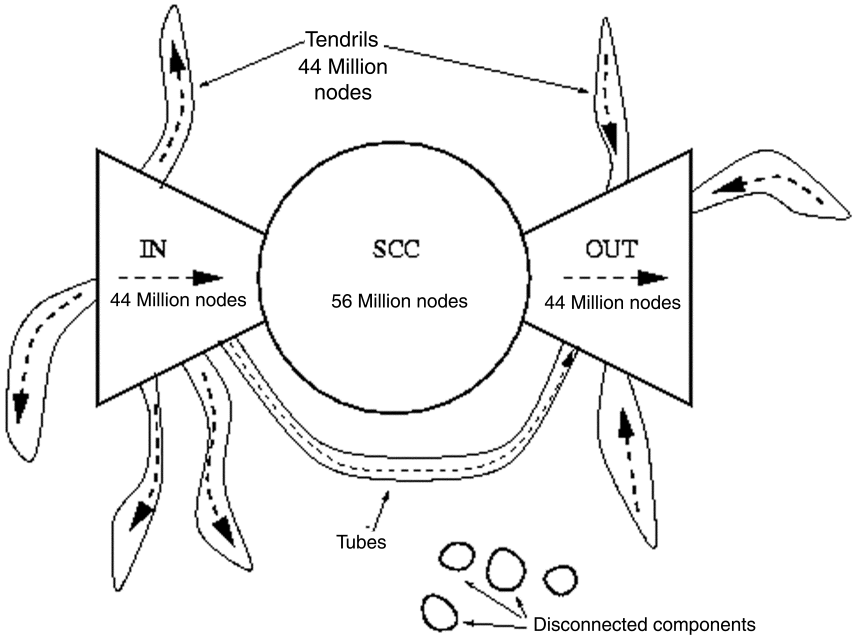
Core: 27%

IN: 21%

OUT: 22%

Tendrils: 22%

Disconnected: 8%



- **Core**: SCC (strongly connected component) – can go from any node to any node via a directed path;
- **IN**: can reach core, but cannot be reached from it;
- **OUT**: can be reached from core, but cannot reach it;
- **Tendrils**: (a) reachable from IN but cannot reach core OR/AND (b) can reach OUT but cannot be reached from it;
- **Disconnected**: no path to core even if direction is ignored.

Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., ... & Wiener, J. (2000). Graph structure in the web. *Computer networks*, *33*(1-6), 309-320.

Why do we Care About the Web Graph?

Exploit the Web structure for:

- Crawlers;
- Search and link analysis;
- Spam detection;
- Community discovery;
- Classification/organization.

Predict the future of the Web:

- Mathematical models;
- Algorithm analysis;
- Sociological understanding;
- New business opportunities;
- New politics.

Largest human artifact ever created (?)

MAIN CHALLENGES OF WEB DATA PROCESSING

Big Data Challenges

Web data: Web pages, blogs, tweets, Amazon clicks, YouTube comments, dr.dk updates, SourceForge downloads, satellite data,...

Big data: term coined by META (now Gartner) in 2001.

"Big Data problem": The rate of data accumulation is rising faster than our cognitive capacity to analyse increasingly large datasets to make decisions.

Challenges:

- Volume, Variety, Veracity, Velocity (the 4 Vs of data);
- Situation, Scale, Semantics, Sequence (the 4 Ss of data).

The 4 Vs of Data: Volume

Volume: substantially large-scale & increasing. Examples:

- 90% of data has accumulated in the last 2 years (Ghavami, 2016);
- Self-driving cars will generate 2PB of data every year;
- The entire writings of all humankind from the beginning of history up to now in all languages is ~50 petabytes;
- Data volume in 2025 will be 175ZB (International Data Corporation).

Gigabyte – 1K Megabytes	A movie of TV quality
Terabyte – 1K Gigabytes	All x-ray films in a large hospital
Petabyte – 1K Terabytes	Half of all US academic research libraries
Exabyte – 1K Petabytes	Data generated from SKA telescope per day
Zetabyte – 1K Exabytes	All worldwide data generated by June 2012
Yottabyte – 1K Zetabytes	$1YB=1000^8$ bytes

The 4 Vs of Data: Variety

Variety:

- Data previously confined to paper are now digital & new forms of data, previously non-existent;
- Both human and machine generated;
- Almost all devices will soon generate their own data: sensors, smart pumps, ventilators, audio recordings of patient-doctor sessions, videos captured during surgery, colour images of wounds, customer sentiment, social media, genetic sequence,...

Internet of Things (IoT): all devices communicate freely with each other through the Internet:

- Heterogeneous: .pdf, .txt, .html, .css, .xml, .gif, .mp3, .mp4,...
- Semi/un/structured: database, excel, free text,...

The 4 Vs of data: Veracity and Velocity

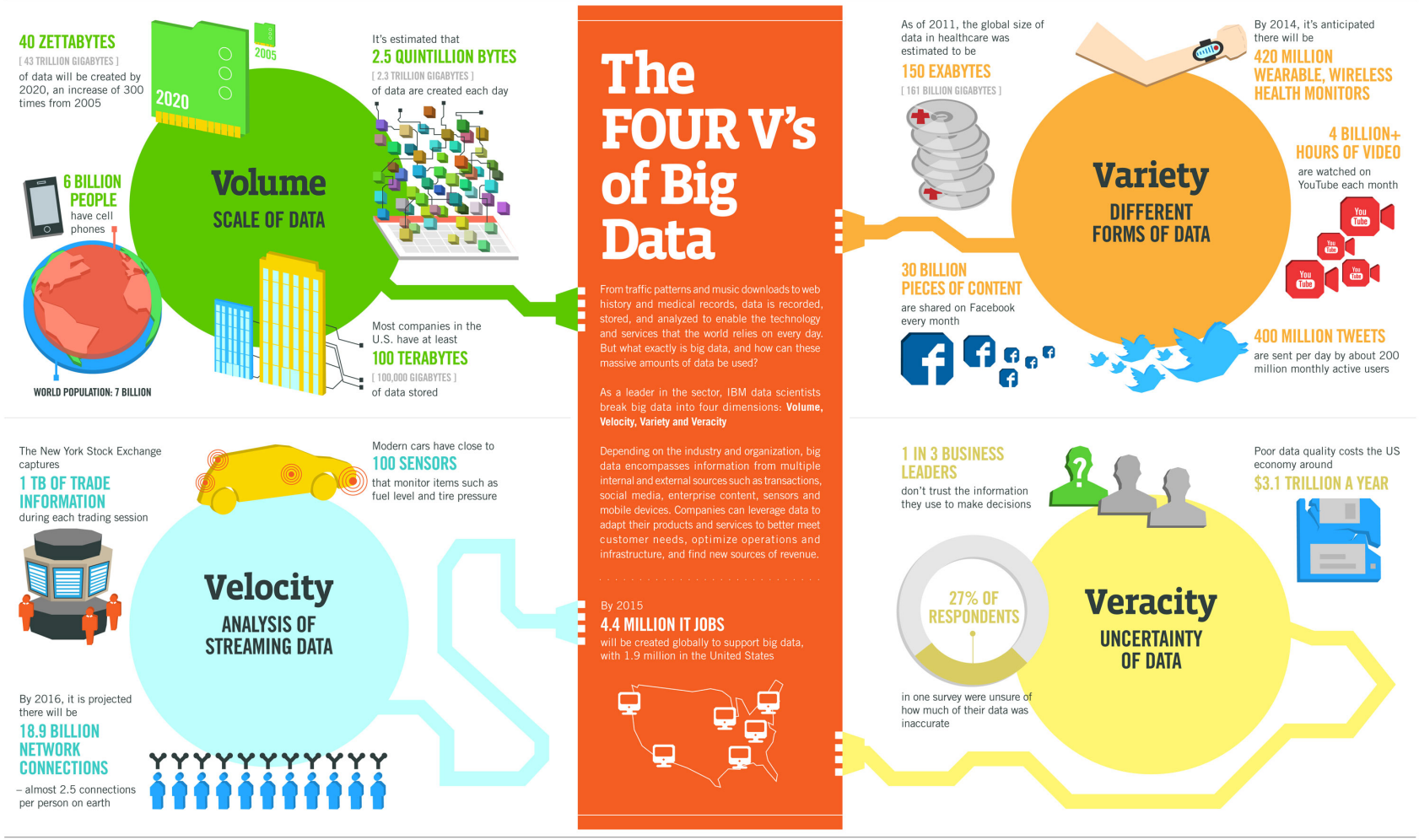
Veracity:

- Uncertainty (unknown/unreliable sources,...)
- Noise-prone: intention (e.g. spam), form (e.g. typos, chat lingo), content (e.g. bias, factual error, fake news VS sarcasm), source (e.g. near-duplicate ~40%, corrupt OCR),...

Velocity:

- Dynamic: updated at various frequencies, often without warning (e.g. no timestamp on the update);
- Time-series: (some): from near real-time (e.g. instant messaging) to periodicity (e.g. Web search query logs).

The 4 Vs of data



https://www.ibmbigdatahub.com/infographic/four-vs-big-data

The 4s of Data

- **Situation**: context of data measurement. E.g. Blood pressure value when at rest VS standing up VS after climbing stairs.
- **Scale**: data with limited range VS wide range. A slight change can be significant in limited range data, but might make sense to ignore in wide range data.
- **Semantics**: circa 80% of data is unstructured → extracting pertinent terms from unstructured data is a challenge.
- **Sequence** (same as Velocity): sequential or time series data.

What is the Size of the Web?

Surprisingly **hard to answer**!

- Naïve solution: keep crawling until the whole graph has been explored;
- Extremely simple but wrong solution: crawling is complicated:
  - Spamming, duplicates, mirrors,…
- Simple example of a complication: Soft 404
  - If a page does not exist, the server is supposed to return an error code = "404";
  - Many servers do not return an error code, but keep the visitor on site, or simply send the visitor to the home page;
  - More on HTTP status codes later.

Estimating the Size of the Web

- The Web that we see is what Web crawlers discover;
- We need large crawls in order to make meaningful measurements;
- The measurements are still biased by:
  - The crawling policy;
  - Size limitations of the crawl;
  - Perturbations of the "natural" process of birth and death of nodes and links;
  - More on crawling later.

Estimates:

1999: 800 million pages (Lawrence and Giles)

2008: 1 trillion pages (https://googleblog.blogspot.dk/2008/07/we-knew-web-was-big.html)

2016: 130 trillion pages (https://searchengineland.com/googles-search-indexes-hits-130-trillion-pages-documents-263378)

The *deep* (or *hidden* or *invisible)* Web contains 400-550 times more information than the known Web [Bergman, 2001: "The Deep Web: Surfacing Hidden Value"]

WEB CRAWLING

What is a Web Crawler?

Crawler (a.k.a. spider, bot, worm, ant, scutter, harvester,…)

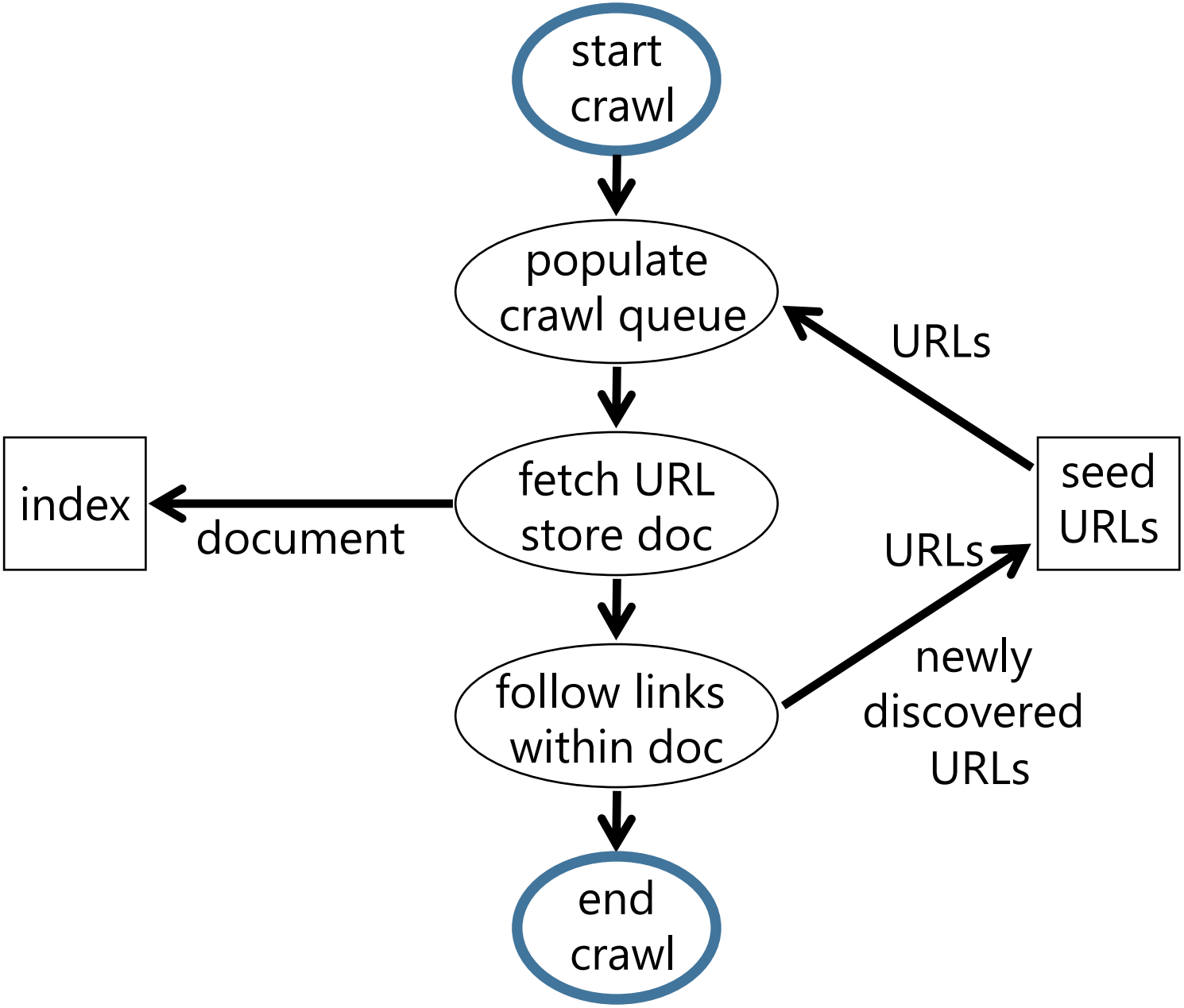
- Program that automatically locates, fetches and stores webpages efficiently & methodically.

Aim: gather as many useful webpages as possible.

How: following hyperlinks:

- Given a list of seed urls (crawl queue);
- Visit each seed url, store its content, & identify all the hyperlinks in it;
- Add the urls of those hyperlinks to crawl queue;
- Repeat until crawl queue is exhausted & subject to policies.

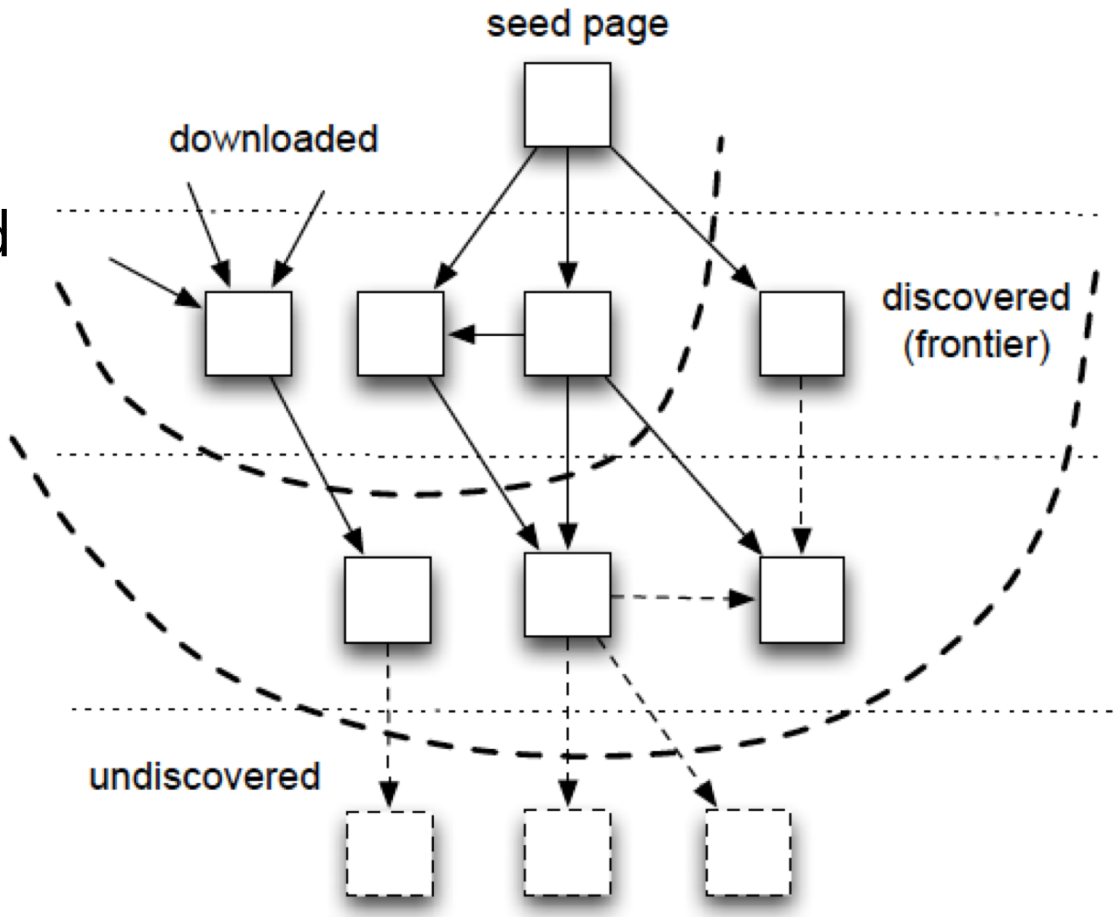
https://developers.google.com/search/docs/advanced/crawling/overview


start crawl
populate crawl queue
URLs
index
fetch URL store doc
seed URLs
document
URLs
follow links within doc
newly discovered URLs
end crawl

Partitioning the Web:

Crawling divides the web into 3 sets:

1. Downloaded
2. Discovered
3. Undiscovered



When to Stop?

Crawl stops when crawl queue is exhausted & **subject to policies.**

Crawling policies:

1. **Selection policy**: which URLs to crawl;
2. **Re-visit policy**: when to re-crawl the same URL;
3. **Politeness policy**: how aggressive the crawl is.

Selection & Revisit: URL Prioritisation Policies

A crawler can only download tiny fraction of Web pages each time, so it needs to prioritise its downloads.

A crawler maintains two separate queues for prioritising the download of URLs:

- **Discovery queue** (selection policy):
  - Downloads pages pointed by already discovered links;
  - Tries to increase **coverage.**
- **Refreshing queue** (re-visit policy):
  - Re-downloads already downloaded pages;
  - Tries to increase **freshness.**

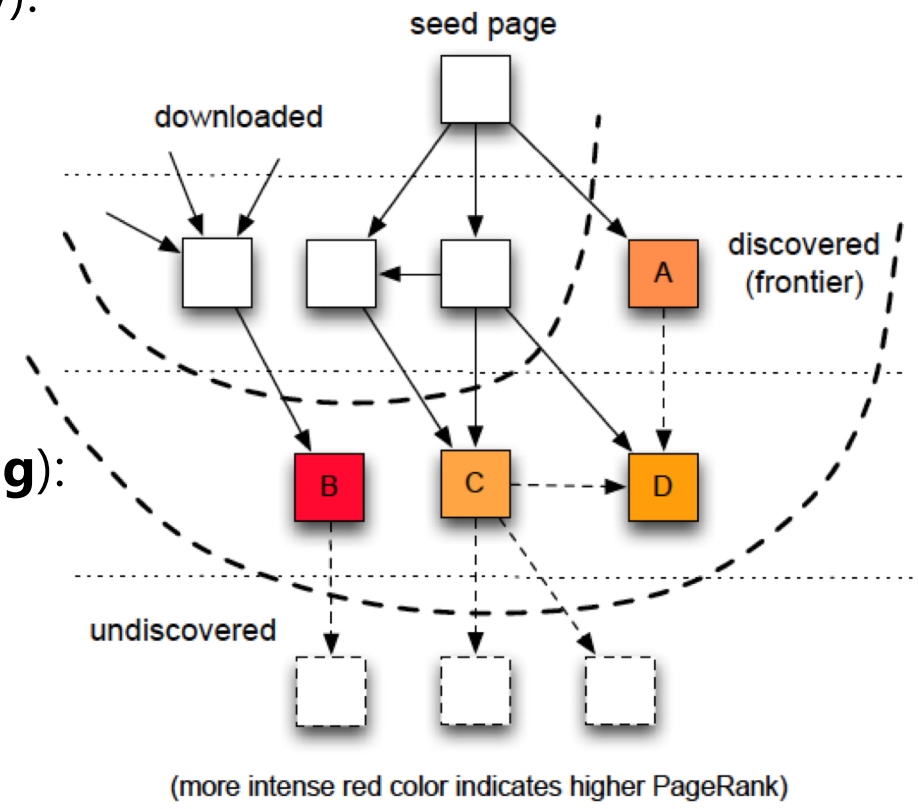
Selection & Revisit Policies

URL prioritisation (**Discovery**):

- Random (A,B,C,D);
- Breadth-first (A);
- In-degree (C);
- PageRank (B).

URL prioritisation (**Refreshing**):

- Random;
- PageRank;
- User feedback/interest;
- Age;
- Longevity.



Why Focus on Discovery and Refreshing?

- Because the Web is **dynamic**: crawling tiny fraction can take months. By the time crawling is done, new information has been added, updated or deleted. Need to re-crawl.
- Because for a search engine there is a **cost** associated with missing webpages or having outdated information:
  - Crawling metrics measure this cost.

Coverage & Freshness are Crawling Metrics

Quality metrics:

- Coverage: % of the Web discovered or downloaded by the crawler;
- Freshness: measure of staleness of the local copy of a page relative to the page's copy on the Web.

Performance metric:

- Throughput: content download rate in bytes per unit of time.

Example of Freshness

Freshness *F* of a webpage *p* stored in the index at time *t* (binary measure) **(1 is best)**

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

Age *A* of a webpage *p* stored in the index at time *t* **(0 is best)**

$$A_p(t) = \begin{cases} 0, & \text{if } p \text{ is not modified} \\ t - \text{modification time of } p, & \text{otherwise} \end{cases}$$

Politeness Policy: how Aggressive the Crawler is?

Why? Crawlers get data very fast & in great depth → **crippling impact on website performance** e.g. if crawler sends multiple requests per sec, or downloads large files, server may not keep up with user requests:

- **Network resources**: considerable bandwidth for a long period of time;
- **Server overload**: if frequency of accesses to server is high;

Poorly written crawlers may crash servers or routers or may download Web pages they cannot handle.

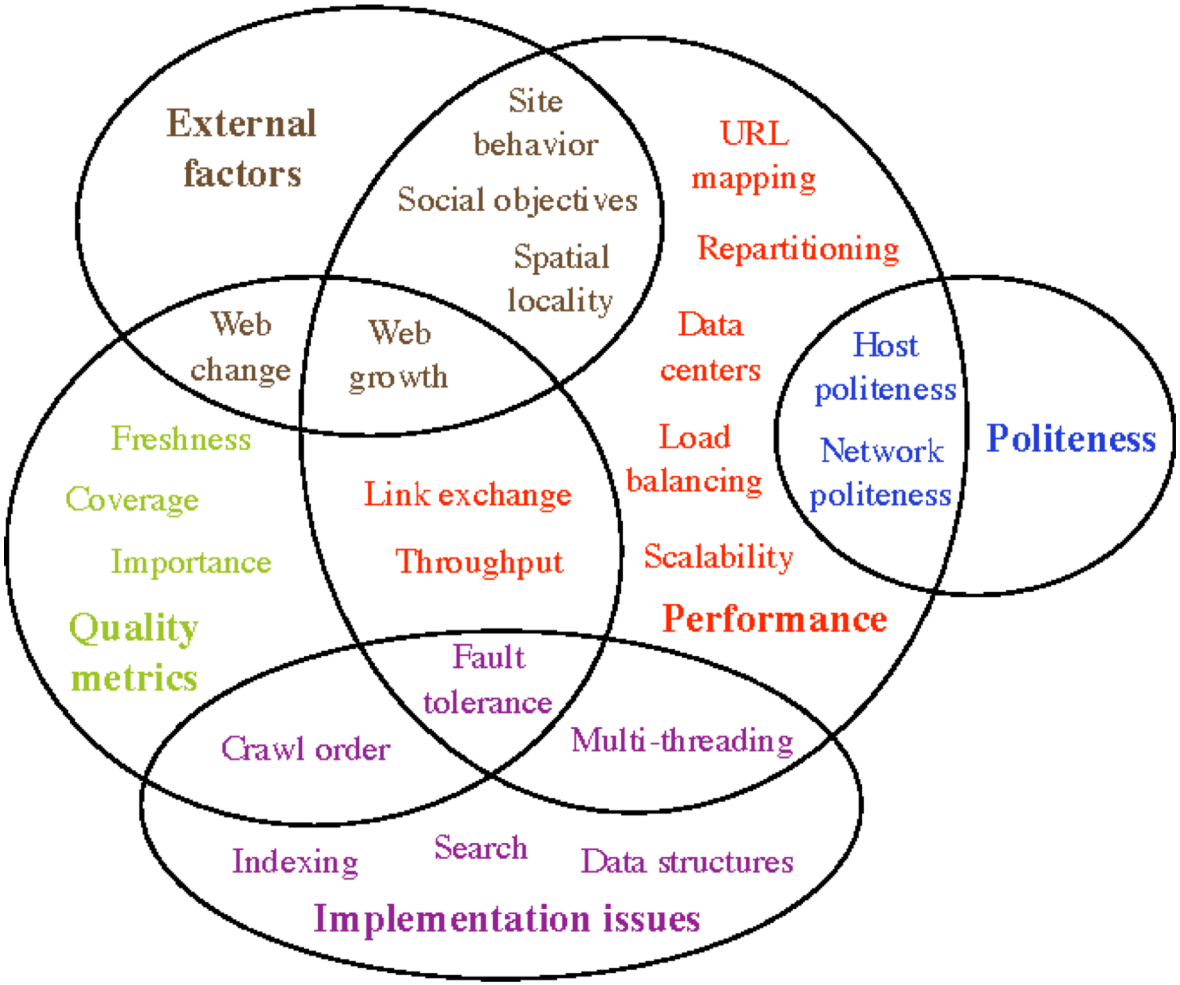
Partial Solutions

- A polite crawler puts a delay between two consecutive downloads from the same server (common: 20 seconds);
- A polite crawler closes the connection after the page is downloaded from the server;
- A polite crawler respects the **robots exclusion protocol**:
  - Created by Web page administrators to indicate which parts of their servers should and/or should not be crawled;
  - robots.txt: standard from the early days of the Web;
  - Crawlers often cache robots.txt files for efficiency.

How Google handles such exclusion protocols:

https://developers.google.com/webmasters/control-crawl-index/docs/robots_txt

Concepts Related to Web Crawling



Source: https://www.semanticscholar.org/paper/On-the-feasibility-of-geographically-distributed-Cambazoglu-Plachouras/b64eff5e64b7a403d60d3aa818c0a895bcfe2bb7

WEB CRAWLING IN PRACTICE

Implementation of Web Crawlers

Crawlers: central part of search engines

- Details of their algorithms & architecture are kept as **business secrets** (lack of detail in published designs).

Why?

- **Competition**: prevent others to reproduce the work;
- **Spamming risks**: emerging concerns about spammers taking advantage of the crawling process to spread spam.

https://www.google.com/search/howsearchworks/crawling-indexing/

Crawling Architectures

- **Single computer**
  - CPU, RAM, and disk becomes bottleneck;
  - Not scalable.
- **Parallel**
  - Multiple computers, single data centre;
  - Scalable.
- **Geographically distributed**
  - Multiple computers, multiple data centres;
  - Scalable;
  - Reduces network latency.

Geographically Distributed Web Crawling

Benefits:

- Higher crawling throughput:
  - Geographical proximity;
  - Lower crawling latency.
- Increased availability:
  - Continuity of business.
- Better coupling with distributed indexing/search:
  - Reduced data migration.

Focused Web Crawling

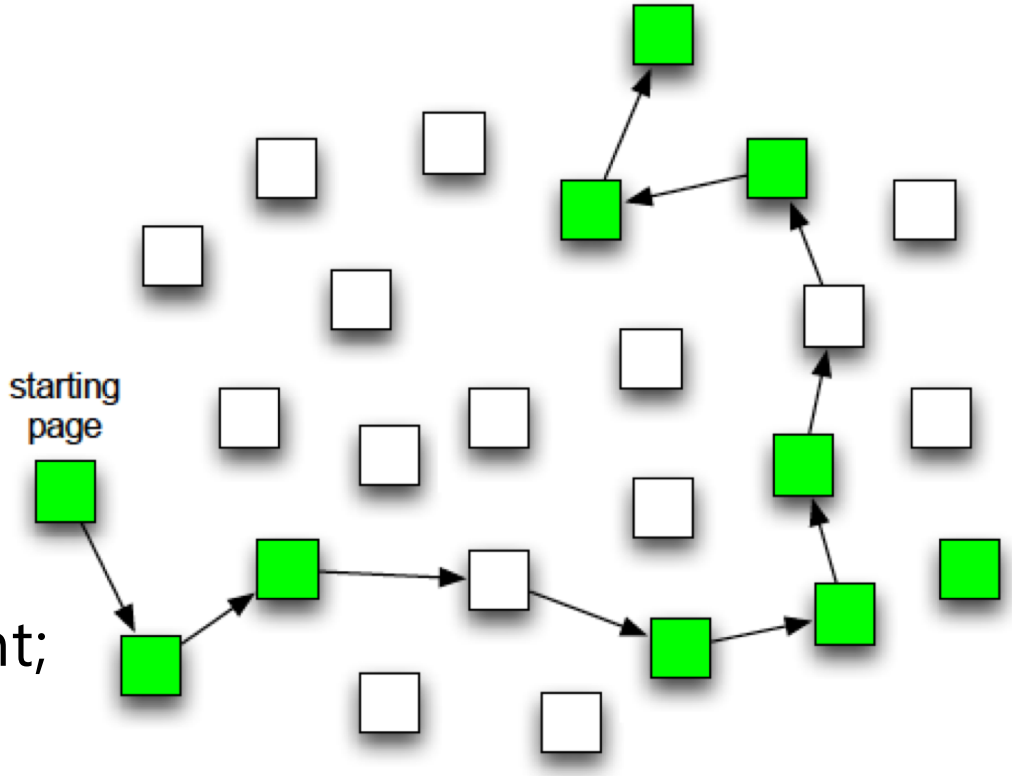
Goal: locate and download a large proportion of Web pages that match a given target theme as early as possible.

Example themes:

- Topic (Covid);
- Media type (forums);
- Demographics (kids).

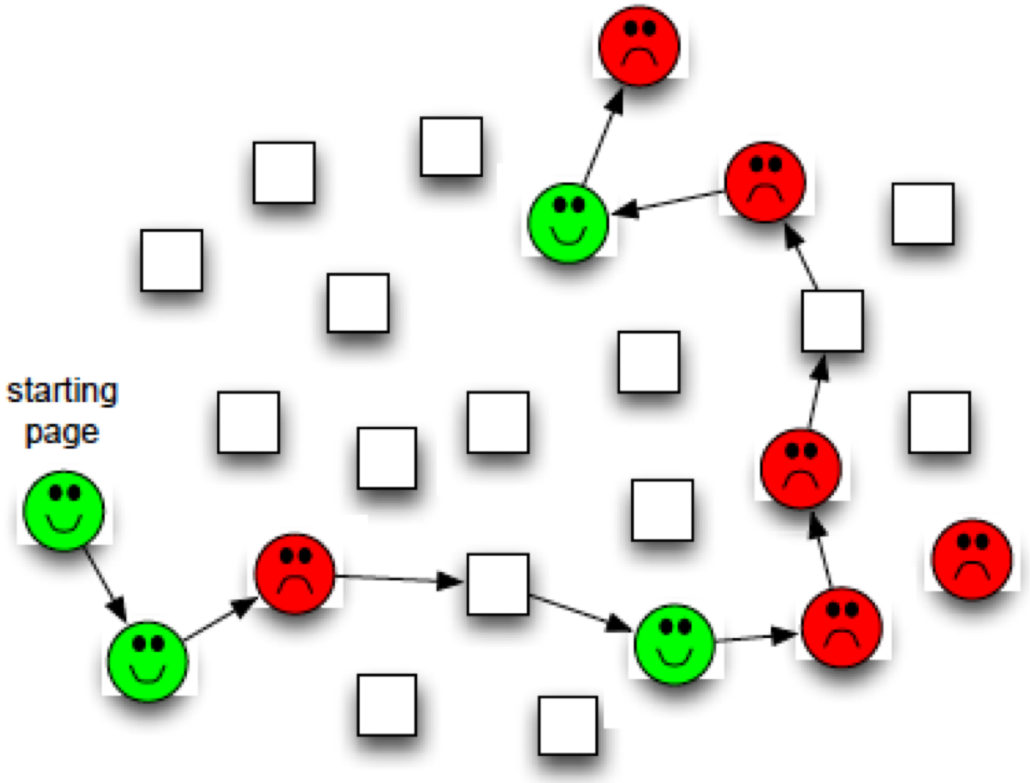
Strategies:

- URL patterns;
- Referring page content;
- Local graph structure.



Sentiment Focused Web Crawling

Goal: locate and download a large proportion of Web pages that contain positive or negative sentiments (opinionated content) as early as possible.



Research Problem: Hidden Web Crawling

Hidden Web: Web pages that a crawler cannot access by simply following link structure:

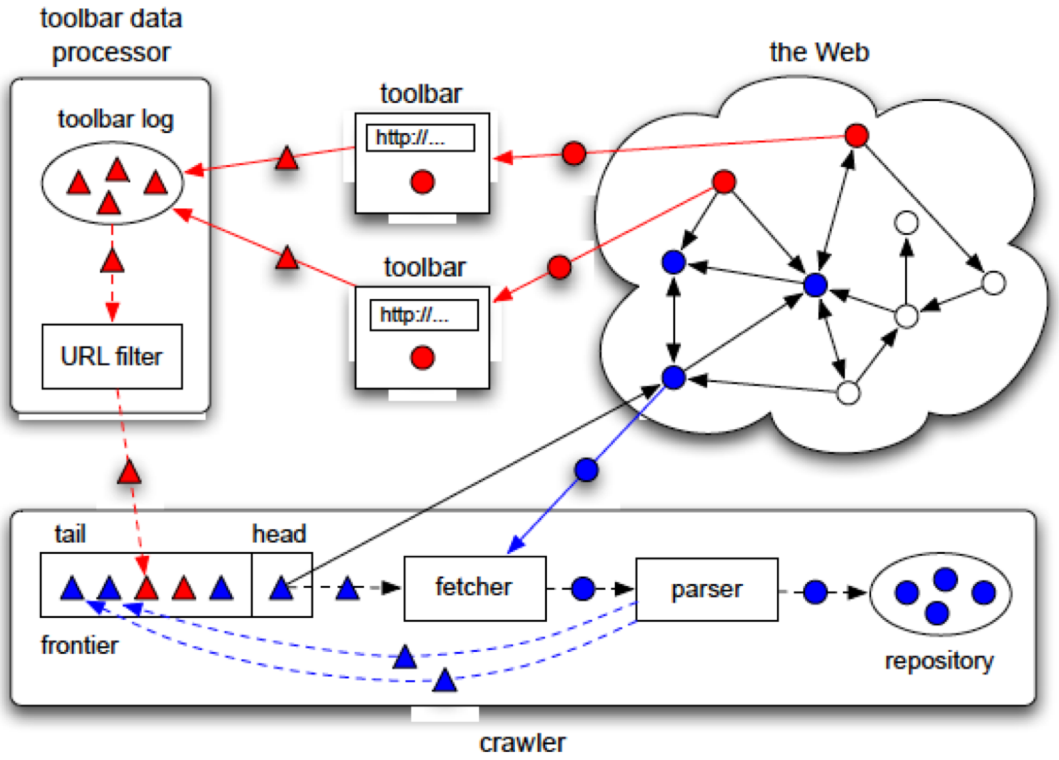
Examples:

- Unlinked pages;
- Private sites;
- Scripted content;
- Dynamic content;
- …

Hidden Web Crawling: Passive Discovery

URL discovery by external agents: toolbar logs, email messages, tweets,...

Benefits: improved coverage, early discovery.



Writing your Own Web Crawler

- Python libraries for crawling:
  - BeautifulSoup: HTML and XML parser, combine with `requests`, `urlib2` to open URLs and store result;
  - Scrapy: Parsing and opening URLs in one go.

- Tutorial: https://www.datacamp.com/community/tutorials/making-web-crawlers-scrapy-python

- Be very careful
  - Observe politeness policies;
  - Websites temporarily ban IPs of those who disregard policies (e.g. Google, ArXiv).

HTTP Status Codes

- *1xx informational response* – the request was received, continuing process;
- *2xx successful* – the request was successfully received, understood and accepted;
- *3xx redirection* – further action needs to be taken in order to complete the request;
- *4xx client error* – the request contains bad syntax or cannot be fulfilled;
- *5xx server error* – the server failed to fulfill an apparently valid request.

- Full list of status codes: https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

More on HTTP Status Codes: 3XX

- 200 OK:
  - The request was successful, the Web page exists.
- 301 Moved permanently:
  - The Web page has been permanently replaced by another page.
- 302 Moved temporarily:
  - The Web page has been temporarily replaced by another URL; users are sent to the redirect's target URL.
- 304 Not modified:
  - The resource has not changed since the last visit (rarely used).
- 307 Temporary redirect:
  - The URL is served over HTTPS rather than HTTP.

More on HTTP Status Codes: 4XX

- 400 Bad request:
  - The server cannot or will not process the request due to an apparent client error (e.g., malformed request syntax, size too large, invalid request message framing, or deceptive request routing).
- 401 Unauthorized:
  - Authentication is required and has failed.
- 404 Not found:
  - The requested resource could not be found but may be available in the future.
- 410 Gone:
  - The resource requested is no longer available and will not be available again.
- 429 Too many requests:
  - The user has sent too many requests in a given amount of time.

More on HTTP Status Codes: 5xx

- 500 Internal server error:
  - Generic / fall-back server error code.
- 501 Not implemented:
  - Server does not recognise the request method or cannot fulfill it (technically).
- 503 Service unavailable:
  - The server cannot handle the request (because it is overloaded or down for maintenance); usually temporary.
- 504 Gateway timeout:
  - The server was acting as a gateway or proxy and did not receive a timely response from the upstream server.

Published Web Crawler Architectures

- Bingbot: Microsoft's Bing Web crawler;
- FAST craweler: Used by Fast Search & Transfer;
- Googlebot: Web crawler of Google;
- PolyBot: a distributed Web crawler;
- RBSE: The first published Web crawler;
- WebFountain: A distributed Web crawler;
- Web RACE: A crawling and caching module;
- Yahoo Slurp: Web crawler used by Yahoo search.

Open Source Web Crawlers

- DataparkSearch: GNU General Public License (GPL);
- GRUB: open source distributed crawler of Wikia Search;
- Heritrix: Internet Archives crawler;
- ICDL Crawler: cross-platform web crawler;
- Norconex HTTP Collector: licensed under GPL;
- Nutch: Apache License;
- Open Search Server: GPL License;
- PHP-Crawler: BSD license;
- Scrapy: BSD license;
- Seeks: Affero GPL.

Today's Lecture

- Course administration

- What is Web Science;
- What is the Web;
- What is the Internet;
- Web graph;
- Main challenges of web data processing;
- Web crawling.

References and Sources:

- Chapter 13 from the book *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. By David Easley and Jon Kleinberg. Cambridge University Press, 2010. Complete preprint on-line at http://www.cs.cornell.edu/home/kleinber/networks-book/
- Chapter 1 from the book *Big Data Analytics Methods: Modern Analytics Techniques for the 21st Century*. By Peter Ghavami. Amazon, 2016.
- S. Lawrence and C. L. Giles. *Accessibility of Information on the Web*. Nature, 400, 107-109, 1999.
- T. Berners-Lee, W. Hall, J. A. Hendler, K. O'Hara, N. Shadbolt and D. J. Weitzner. "*A Framework for Web Science*", Foundations and Trends® in Web Science: Vol. 1: No. 1, pp 1-130, 2006.
- All pictures retrieved with Google for noncommercial reuse

Seminal Readings on Crawling

- Cho, Garcia-Molina, and Page, "Efficient crawling through URL ordering", WWW, 1998.
- Heydon and Najork, "Mercator: a scalable, extensible web crawler", WWW, 1999.
- Chakrabarti, van den Berg, and Dom, "Focused crawling: a new approach to topic-specific web resource discovery", Computer Networks, 1999.
- Najork and Wiener, "Breadth-first crawling yields high-quality pages", WWW, 2001.
- Cho and Garcia-Molina, "Parallel crawlers", WWW, 2002.
- Cho and Garcia-Molina, "Effective page refresh policies for web crawlers", ACM TDS, 2003.
- Lee, Leonard, Wang, and Loguinov, "IRLbot: Scaling to 6 billion pages and beyond", ACM TWEB, 2009.

Thank you!