Web Sampling:
Some theory and Practice

Dr Stephen Wattam @StephenWattam
Dr Paul Rayson @perayson
Andrew Moore @apmoore94

School of Computing and Communications
Lancaster University

27-06-2017

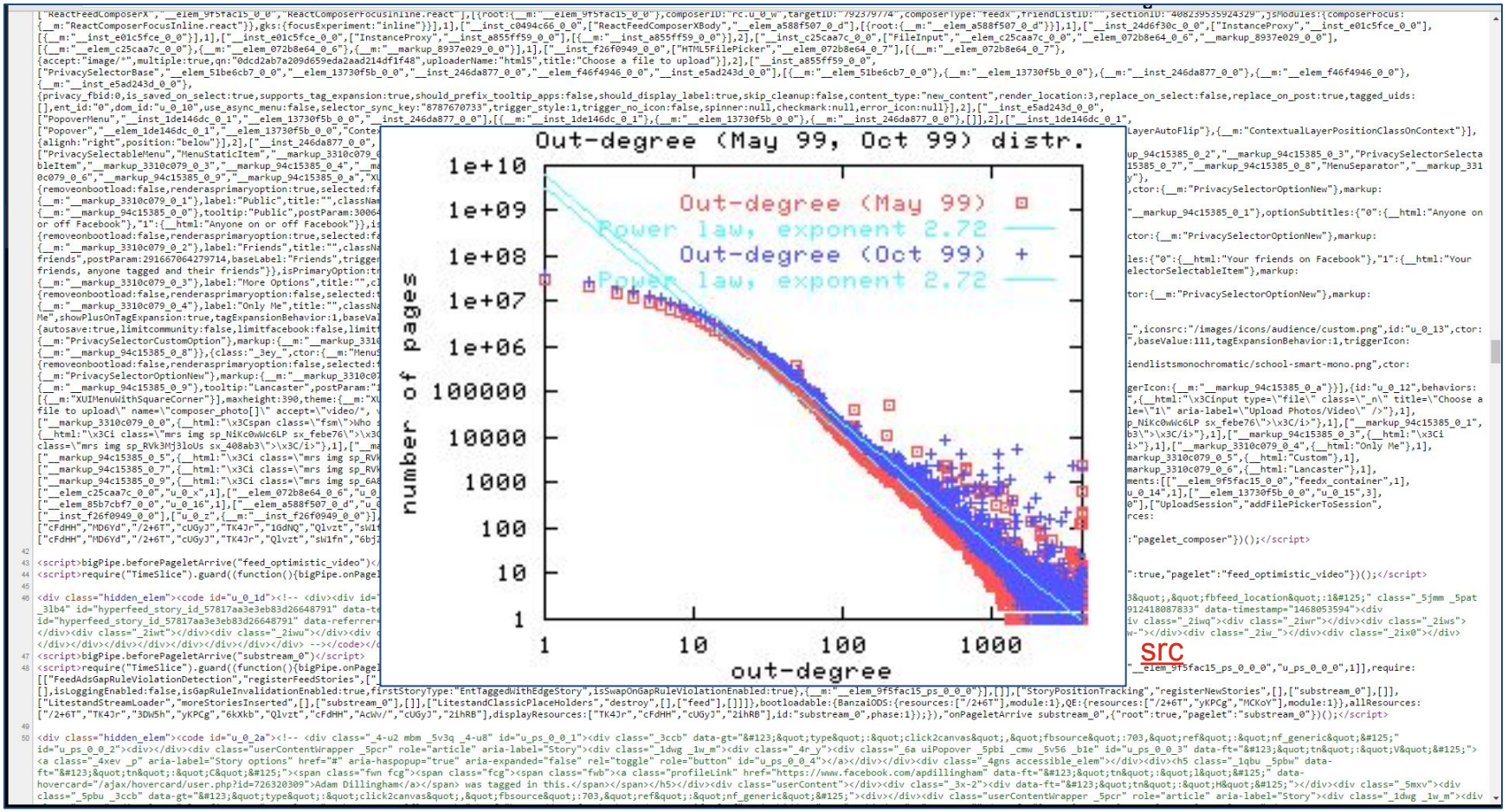
Lancaster University

Overview of Scraping Methods

Sampling

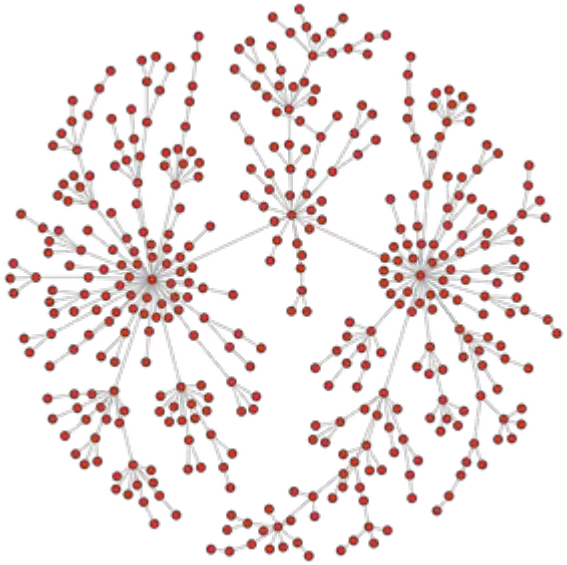
1. Identify the purpose of the sample
2. Identify available variables
3. Enumerate population
4. Define sampling frame w.r.t. external variables
5. Retrieve data
6. Operationalise data

Sampling Online Data

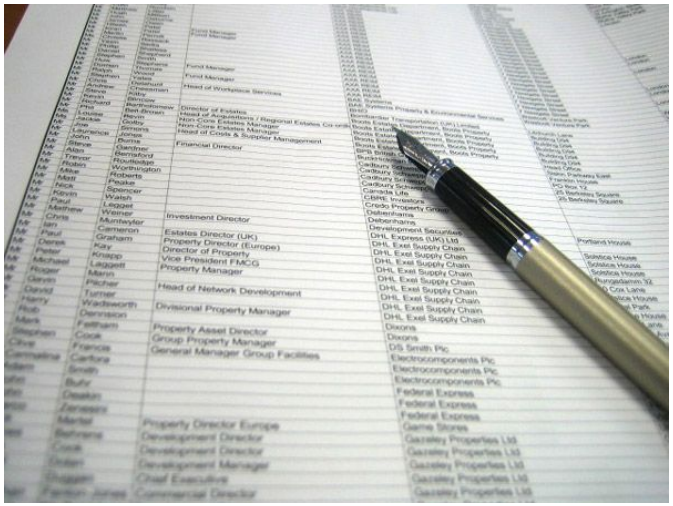


Approach

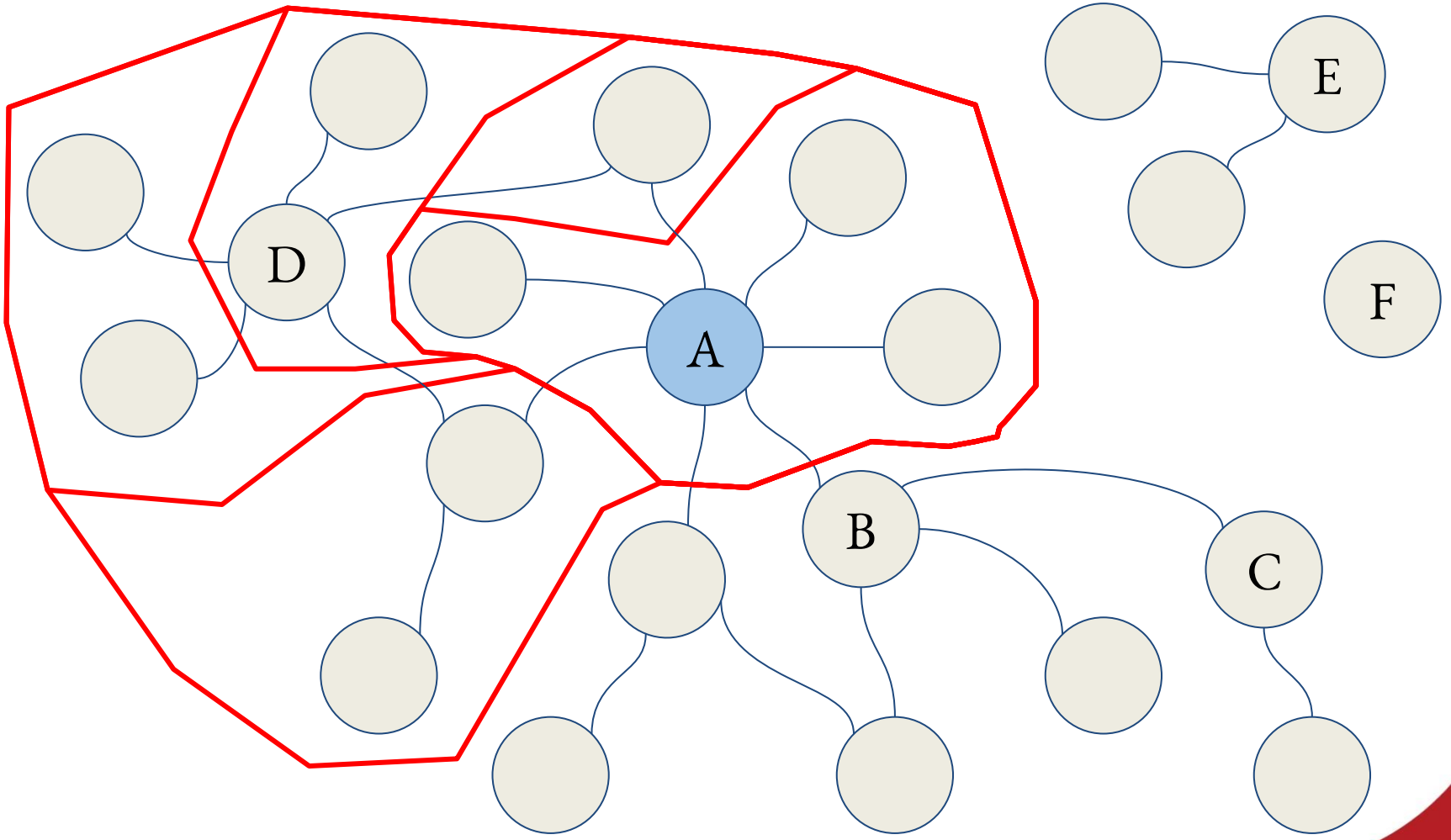
Browsing



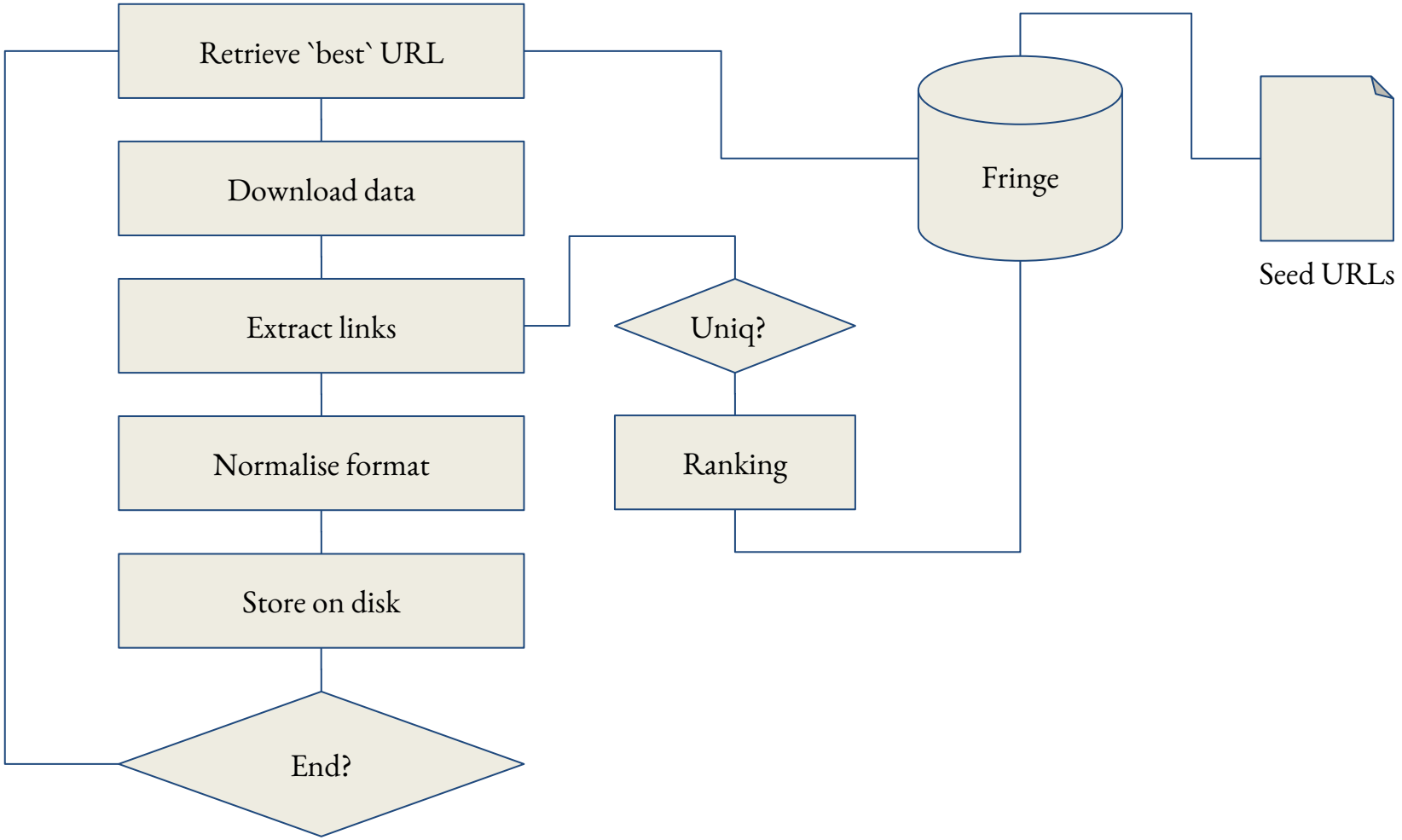
Searching



Approach



Overview of Implementation



Implementation Choices

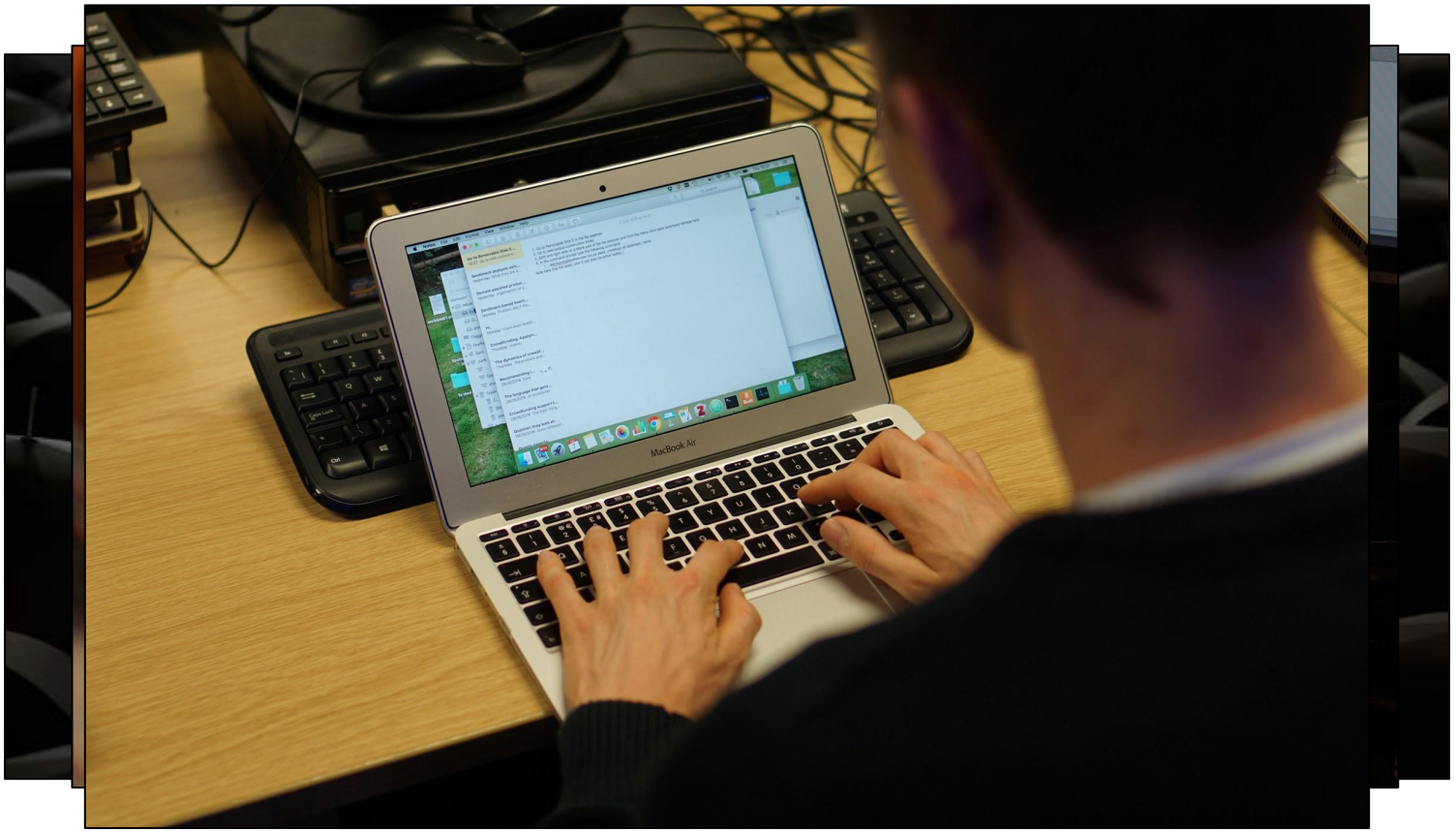
- Parallelisation
- Distribution
- Data Storage
- Throughput (network)
- Deduplication
- Data handling (charset, MIME)
- Consistency

Choosing a ranking (fitness) function

Estimate $\pi(x_i)$ for all pages

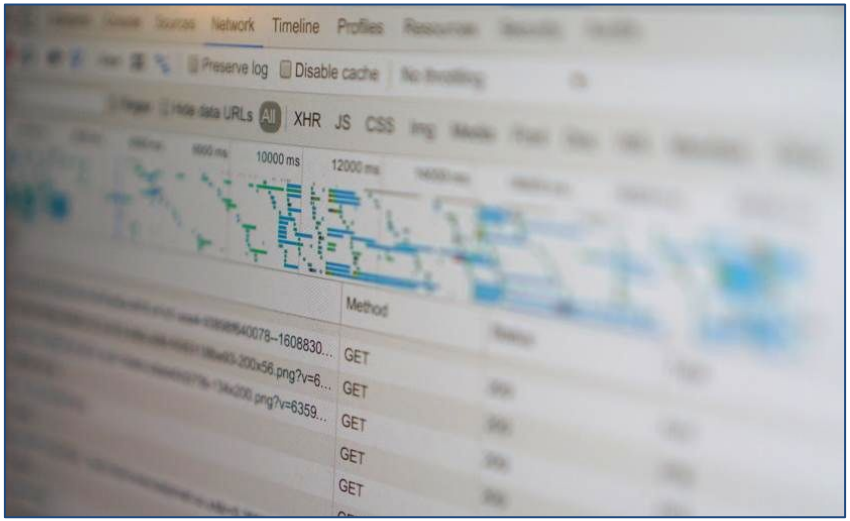
- Based on RQs
- Mindful of network topology
- Use only external variables
- Incorporate other priorities

Your Tasks

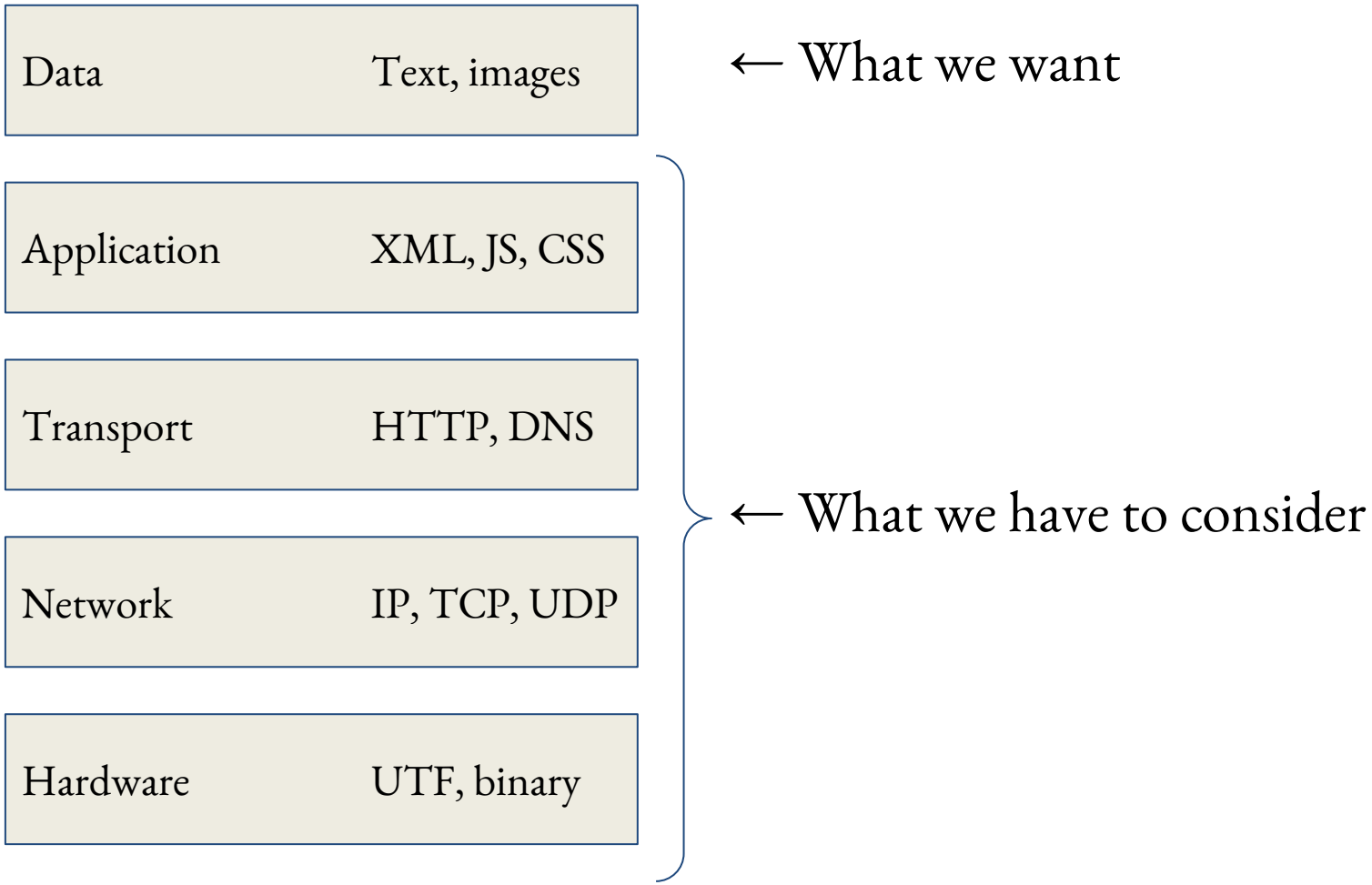


Lancaster University

Complexities of Web Data



Web Tech Stack



Lancaster University

P1: Identify Technical Challenges

- Open your URL lists
- Visit some of the URLs
- Identify which aspects of the data will cause problems
  - Are all of the URLs functional?
  - What kind of data do they return?
  - How will you get metadata?
  - Is this relevant to your RQs?

Technical Challenges

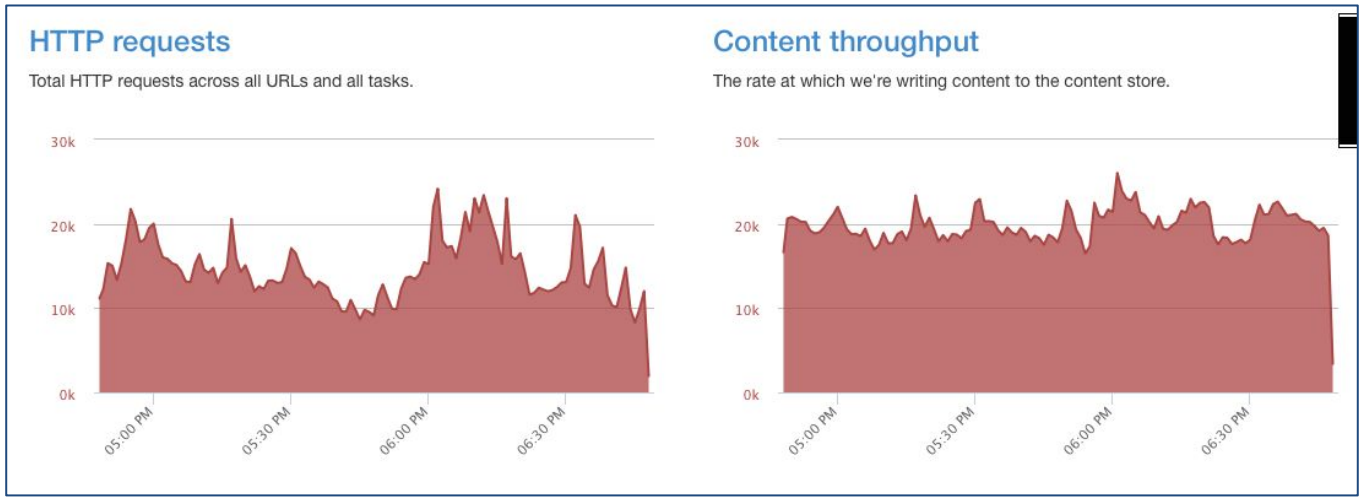
- Encoding
- File type (images, PDF, etc)
- File size
- Timeouts
- Error pages/codes
- Metadata availability
- Metadata normalisation
- Inaccurate metadata
- Boilerplate
- De-duplication (page, url)
- Request headers
- Logins
- Form submission
- AJAX
- Media
- Javascript

Less Technical Challenges

- Content changes
- Active pages
- Session-based content
- UA-based content
- Geographic availability
- Maintenance periods
- Closed communities
- Assumptions of privacy
- Rehosted content
- Personally identifiable data
- Network topology
- Links that change content

Etiquette

src



src

Solution – Careful Engineering

- Check for and handle common edge cases
- Fail gracefully
- Idempotent behaviour if possible
- Keep everything
- Log everything
- Be nice to hosts

Lancaster University

Implementation

Design

Retrieval	Python	Native (urllib2)
Storage	SQL metadata, page data on disk	Sqlite3 & native
Metadata extraction	HTML Parser, HTTP metadata	BeautifulSoup & native
Deduplication	Fuzzy hashing	ssdeep
End condition	URL count, depth, time	Native
Fitness	URL complexity, similarity to current data	Native

Design II: Return of the Design

1. Select a URL
2. Retrieve from web
3. Extract features
4. Accept/reject page
5. Accept/reject links
6. Rank links
7. Write data to db/disk
8. End?

Design III: This time it's Pluggable

Filter.py

`accept(body, metadata)`

Return a Falsy value to reject the current page

URLFilter.py

`accept(url)`

Return a Falsy value to prevent crawling of a URL

URLRank.py

`goodness(url)`

Return a numeric value indicating how to rank URLs. High = good, low = bad

EndCondition.py

`end(corpus_table, body, metadata)`

Return Truthy value to exit, Falsy to continue spidering

Lancaster University

P2: Fitness Function, End Condition

- Open and edit the fitness function in **SampleURLRank.py**
- Open and edit the end condition in **SampleEndCondition.py**
- Optionally fiddle with the filters in **.*Filter.py**

- Stick to simple logic, make few assumptions about the page
- Look for unintentional side-effects

Stuff we didn't do

Fuzzy deduplication

Interpreting JS, DOM changes

Cookies

UA/referer spoofing

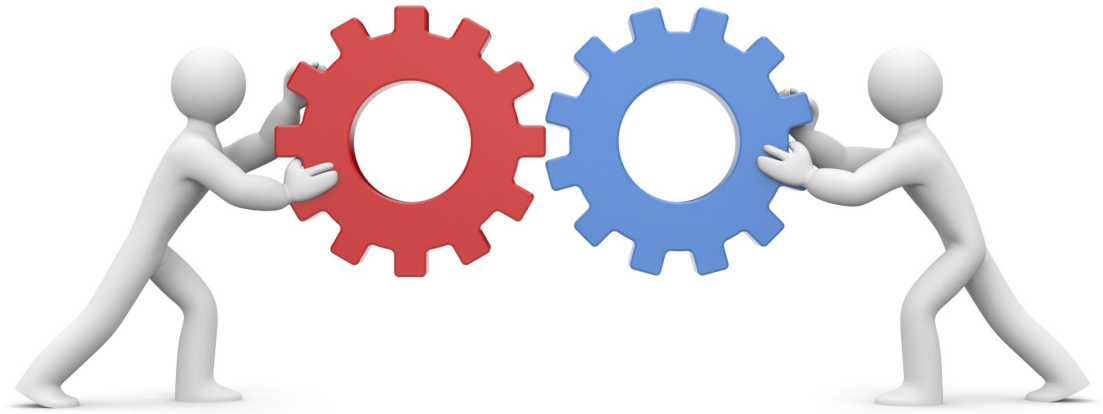
Distribution

robots.txt checks

Error checks, log analysis

Lancaster University

Epilogue:
Using Pre-Defined APIs



Advantages of API Use

Ease of Development

Politeness

Linked data & Metadata Richness

Documentation

Speed (sometimes!)

Disadvantages of API Use

Data source lock-in

Data provider controls your access

Not representative of end-user access

Cost

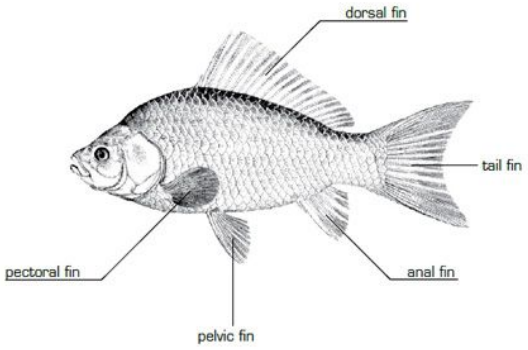
Speed (sometimes!)

When to use APIs?

- Is all data accessible from one API?
- Are the metadata particularly important?
- Can the data be traversed efficiently using the API?
- Is the cost prohibitive?
- Will rate limits make access impossible within a sane timescale?

Lancaster University

Fin.



Resources

- Web Corpus Construction (right)
- Know thy enemy: Web tech RFCs
  - w3.org
  - ietf.org
- Various scraping libs:
  - mechanize, beautifulsoup, scrapy, selenium
- Scrapy documentation on selectors
- Graph theory: Stanford, scale-free network models