LWAC: Longitudinal Web as Corpus

Stephen Wattam, Paul Rayson, Damon Berridge

Lancaster University

July 22, 2013
Rationale
  Rationale
  Sample Design

LWAC
  Design
  Properties
  Implementation
  Workflow

Performance & Limits
  Throughput
  Resources/Scalability

Summary
**WaC for Language Change**

Many ways of measuring change online:

- Crawlers/Revisiting
- Diachronic corpora
- Monitor Corpora
- Subsampling supercorpora
- Feed corpora
I S S U E S

- Irregular visits to pages
  “...visiting each website again in the next crawl anticipating for new content is cost-inefficient.”

- Manual supervision required

- Lack of detail on network properties

- Non-versioned corpus formats
Design Principles

- Reliable, regular sampling strategy
- Set and forget operation
- Vertical and horizontal comparability
- Rigorous & exhaustive data collection
- Integrated corpus format
COHORT SAMPLING

- Common longitudinal design
- Used elsewhere to disambiguate long- from short-term effects
- Fits with open-source, URL-based corpus model
**Cohort Sampling**

![Diagram of cohort sampling](image)

- **Sampling Frame**
- **Time**
- **Website URIs**
- **Batch**
- **Crawl**
Cohort Sampling

Sampling Frame

<table>
<thead>
<tr>
<th>Time</th>
<th>Website URIs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Batch
Crawl
Cohort Sampling

Sampling Frame

Time

Website URIs

Batch
Crawl
USES

Observing network properties over time:

- Link rot/document attrition
- Latency
- Server properties, headers, protocol support

Observing user’s experience of common websites/links:

- Editorial policy
- Page revisions
- “live” page content
LWAC

- Download/Sampling tool for longitudinal use
- Suitable for long- or short-term samples
- Reliable
- Scalable
- Hard to detect
- Records network and content related variables
DATA

- Vertical and horizontal comparability of samples
- Configuration, network properties and data recorded for later use
- ‘No data left behind’ policy: \(\approx 120\) variables stored on each request
- Format, size filters to exclude unwanted data
- Charset normalisation
Import/Export

- Import URL lists
- Export to CSV, XML or arbitrary templates
- Export using filters and data normalisation scripts
- Live export supported
USER MODEL

LWAC can imitate real users or crawlers:

- Realistic redirect handling
- Timeouts at all stages of URL lookup
- Spoofing of user-agent
- Realistic request headers, cookie use
RELIABILITY

- No skew on sampling intervals
- Data security across crashes/restarts (atomicity)
- Error reporting, detailed logging
- Stability for long-term runs
OVERVIEW

- UNIX-model tool set
- Written in Ruby using cURL
- Distributed client-server design
- Central control of sampling policy
**Basic Workflow**

1. Find links of interest ([Web]BootCaT)
BASIC WORKFLOW

1. Find links of interest ([Web]BootCaT)
2. Import links (lwac import)
**Basic Workflow**

1. Find links of interest ([Web]BootCaT)
2. Import links (`lwac import`)
3. Set up clients (`lwac client`)
**Basic Workflow**

1. Find links of interest ([Web]BootCaT)
2. Import links (`lwac import`)
3. Set up clients (`lwac client`)
4. Run server (`lwac server`)
**Basic Workflow**

1. Find links of interest ([Web]BootCaT)
2. Import links (`lwac import`)
3. Set up clients (`lwac client`)
4. Run server (`lwac server`)
5. Drink coffee (RFC 2324)
**BASIC WORKFLOW**

1. Find links of interest ([Web]BootCaT)
2. Import links (`lwac import`)
3. Set up clients (`lwac client`)
4. Run server (`lwac server`)
5. Drink coffee (RFC 2324)
6. Export data (`lwac export`)
**Basic Workflow**

1. Find links of interest ([Web]BootCaT)
2. Import links (`lwac import`
3. Set up clients (`lwac client`
4. Run server (`lwac server`)`
5. Drink coffee (RFC 2324)
6. Export data (`lwac export`)`
7. Do science

![Diagram showing disk, SQL, storage manager, workers, and internet connections]
PERFORMANCE

Dependent on:

- Number of clients
- Number of connections per client
- Client-server network speeds
- Latency/speed of DNS, web servers
THROUGHPUT (1 CLIENT)

Connections vs. time (n=10000; w=1; nginx with 140kb html)
**THROUGHPUT (n CLIENTS)**

Download times for n clients (n=10k, real-world data)
THROUGHPUT (REAL-WORLD)

Corpus:

- BootCaT-derived URL List
- 228k URLs: 4600 requests, 50 links/call

System:

- 3 clients, 400 connections/client

Throughput:

- 14.9GB in 45 minutes (5.6MBps, 300k links/hour)
- \( \approx \) 588 million words after cleaning
THROUGHPUT (REAL-WORLD)

Download rates for real-world corpus (3 clients)
DOWNLOAD TIMES (CORPORA)

Using my 3-client deployment:

- BE06: a few seconds
- BNC: 8 minutes
- ukWaC: 2.5 hours (or 17 hours before filtering)
- DECOW2012: 12 hours (words); 24 hours (documents)
RESOURCES/SCALABILITY

- Memory usage $O(1)$ for client and server
- Memory usage defined by batch size:
  - Server:
    $$(\text{clients} \times \text{batch.size} \times \text{link.size}) + (\text{batch.size} \times \text{max.resource.size})$$
  - Client: $\text{in.progress} \times \text{max.resource.size}$ (using disk cache)
- Disk usage $O(N)$ for server, $O(1)$ for client.
- Practically around 120-200MB for the application, 1-200MB for data.
ETIQUETTE

- LWAC is capable of DDOS-style throughput
- Normally lists of links contain references to each server a few times
- Within-sample rate controlled by the parallel connection limit
- Between-sample rate defined by sample period
## SUMMARY

- LWAC makes longitudinal sampling easy (ish!)
- Records many more variables than most download systems
- Modest resource requirements
- Fast and scalable
- Fully documented, open source
THE LAST SLIDE

http://ucrel.lancs.ac.uk/LWAC/

Suggestions/comments/bug reports welcome!

s.wattam@lancaster.ac.uk