Common Crawl: Data Collection and Use Cases for NLP

Sebastian Nagel
sebastian@commoncrawl.org

HPLT & NLPL Winter School on Large-Scale Language Modeling and Neural Machine Translation with Web Data, Feb 6–8, 2023
About Common Crawl

- we’re a non-profit that makes web data accessible to programmers and data scientists
- hosted as Open Data set on Amazon Web Services \([1, 2]\)
- for natural language processing, web science, information retrieval, semantic web, internet security research, ...
- since 2008
  - 240 billion web pages (HTML) captured
  - 70 billion unique URLs
  - 7 PiB of data
    - 5 PiB WARC and ARC files
    - 2 PiB derivative data
- sample crawls, not a comprehensive crawl
Data Collection – Web Crawler

Data Collection
Web Crawler 2008 – 2012
Seed Donations 2013 – 2018
Web Crawler 2013 – now
Nutch Default Crawler Workflow
Nutch at Common Crawl 2013 – 2016
Nutch at Common Crawl 2017 – now
Crawler Politeness
Crawler Politeness: robots.txt example
Crawler Politeness Implications
Nutch at Common Crawl – Good to know
Nutch at Common Crawl – Hadoop and hardware
Nutch at Common Crawl – Fetch list layout
News Crawler

Data Collection – Link Prioritization

Data Collection – Representativity, Geographical and Language Bias

Data – WARC and ARC Files

Data – Derivative Formats

Data – Usage

NLP Usage Examples

Summary
Data Collection

data is collected by a web crawler

- polite, respects robots.txt
- open source

three phases of data collection using different

- crawler implementations
- approaches to find and sample (prioritize) seeds and URLs

<table>
<thead>
<tr>
<th>crawler</th>
<th>seeds / link prioritization</th>
</tr>
</thead>
<tbody>
<tr>
<td>2008–2012</td>
<td>in-house pagerank</td>
</tr>
<tr>
<td>2013–2016</td>
<td>Nutch blekko seed donation</td>
</tr>
<tr>
<td>2017–now</td>
<td>Nutch harmonic centrality</td>
</tr>
</tbody>
</table>
- in-house development by Ahad Rana [3, 4]
- batch-based (Hadoop Map-Reduce)
- pagerank calculation
- deduplication as post-processing step
- yearly data releases (after months of crawling)
  - 3 datasets, ARC file format
  - 130 TiB, 8.5 billion page captures, 6.7 billion unique URLs

Common Crawl will use blekko’s metadata to improve its crawl quality, while avoiding webspam, porn, and the influence of excessive SEO [6]

- focus on efficiently and politely fetching web pages
- no need to maintain a large URL frontier and to “steer” the crawl
- more frequent, ideally monthly data releases

2016–2018 occasional seed donations

- up to 400 million URLs
- not enough to “feed” the crawler
Apache Nutch [7, 8, 9]

- scalable, batch-based (Hadoop Map-Reduce)
- extensible and modular (primary focus: feed search index)
- open source, community-based
- with few CC-specific modifications and extensions [10]
- used most notably as efficient, distributed but polite fetcher
Nutch Default Crawler Workflow

0. initialize CrawlDb (aka. “frontier”), inject seed URLs
repeat generate-fetch-update cycle $n$ times:

1. **generate** fetch list(s): select URLs from CrawlDb for fetching
2. **fetch** URLs from fetch list(s)
3. **parse** documents: extract content, metadata and links
4. **update** CrawlDb (fetch status, score, signature), add links

inlined or at the end of one crawler run (once for multiple cycles):

5. **invert links**: map anchor texts to documents the links point to
6. **deduplicate** documents by signature
7. **index** document content, meta data, and anchor texts
8. (data exports)
0. convert list of donated URLs to CrawlDb
1. generate fetch lists
2. fetch URLs
8. export of content and capture metadata as WARC files
4. update CrawlDb with status from preceding crawl (no link additions)
6. flag dups in CrawlDb (by signature and redirect target)
0. inject URLs
  - links sampled from preceding crawl (≈4 billion)
  - “fresh” links from shallow, priority-first crawl (≈1.5 billion)
    - seeded with homepages of 40 million host/domain names
    - 9 cycles in one day
  - URLs sampled from 20 million sitemaps [11] (≈4 billion)
1. generate fetch list(s)
2. fetch URLs and write WARC files
Crawler Politeness

- slow crawling
  - (current configuration) min. 5 seconds between successive requests to the same host
  - further slow down (exponential backoff) if host responds with errors
- respect robots.txt rules and
- related metatags and attributes (eg. nofollow)
- CCBot identifies itself
  - send user-agent string and contact information along with requests
  - no (residential) proxies
Crawler Politeness: robots.txt example

- **User-agent: Googlebot-News**
  Disallow: /angebote/

- **User-agent: ***
  Disallow: /zeit/
  Disallow: /templates/
  Disallow: /hp_channels/
  Disallow: /send/
  Disallow: /suche/
  Disallow: /rezepte/suche/
  Disallow: */comment-thread?
  Disallow: */liveblog-backend*
  Disallow: /framebuilder/
  Disallow: /campus/framebuilder/
  Disallow: /cre-1.0/tracking/*.js$

- **User-agent: Baiduspider**
  Disallow: /

- **User-agent: Applebot**
  Allow: /
  Disallow: /cre-1.0/

- **User-agent: GrapeshotCrawler**
  crawl-delay: 3

- **Sitemap: https://www.zeit.de/gsitemaps/index.xml**

- **https://www.zeit.de/robots.txt**
- Googlebot-News and Applebot ev. preferred (more paths allowed)
- Baiduspider penalized
- GrapeshotCrawler [12] to wait 3 seconds between requests
- default rule set excludes templates, duplicated dynamic content or user comments
- improve quality of crawled content and search results!
- announced sitemap provides an up-to-date list of crawlable URLs
Crawler Politeness Implications

- (with well-written robots.txt) less of
  - private / personal content
  - duplicated content
- significant parts of the web (eg. social media) are not included
- links in disallowed content are not visible to the crawler
- easy-to-adapt way to opt out from being crawled by "CCBot" (or to opt in if default rules disallow crawling)

- crawler architecture is largely determined by politeness
  - proper queuing to guarantee fetch delays
  - minimize efforts to fetch, parse and cache robots.txt
- fetcher job writes WARC files after shuffling the captures
every WARC file is a (pseudo-)random sample by its own
- 1 MiB content limit – longer payloads are truncated
- since Aug 2018: crawler identifies content language via CLD2
- Hadoop cluster running Apache Bigtop
- utilize cheap AWS EC2 spot instances
- fetching: 20 nodes (2 cores, 32 GB RAM)
- Nutch data structures persisted on S3
  $40 per month to store CrawlDb of 25 billion URLs (1.6 TiB)
100 segments, processed sequentially over 14 days, each segment with
40 partitions (one partition is pass to one fetcher task)
  - by host: all URLs of one host are in this partition
  - weakly by domain: hosts of one domain are likely contained in one partition
every partition is shuffled (URLs of the same host are distributed randomly)
keep a certain number of URLs from one host/domain in one segment/partition
  (reduce costly DNS lookups and robots.txt processing/caching)

fetch list: 4 billion URLs from 45 million domains, 60 million hosts
3 billion URLs successfully fetched from 35 million domains, 45 million hosts
reasons for fetch failures: HTTP status other than 200, robots.txt denied, network issues, crawler node failures (using cheap but unreliable spot instances), dropped from fetch list
News Crawler

- since 2016, continuously released [13, 14]
- monthly crawl and release schedule not optimal for the news genre
- crawler follows news feeds and sitemaps
- StormCrawler [15] – “streaming” crawler follows links more quickly (no wait for next batch)
Data Collection – Web Crawler

Data Collection – Link Prioritization

Prioritization – Which Pages or Sites to Crawl

Link Prioritization – Web Graphs and rankings based on Common Crawl

Link Prioritization – Graph-based ranking example

Prioritization – A Deeper Look into the Current Implementation

Data Collection – Representativity, Geographical and Language Bias

Data – WARC and ARC Files

Data – Derivative Formats

Data – Usage

NLP Usage Examples

Summary
Prioritization – Which Pages or Sites to Crawl

Why prioritization is necessary? Why not just follow links?

- an average “monthly” crawl includes 3 billion page captures with
  500+ billion links
  25+ billion unique URLs linked
- a single sitemap (sitemap index) may list up to 2.5 billion URLs
- need to select a representative sample of web pages
- given limited resources and the requirements reg. crawler politeness
2013—2015  Web Data Commons, University of Mannheim: hyperlink graphs and rankings [16, 17, 18]
  - page/host/domain-level hyper-link graphs
  - host-level site ranking by harmonic centrality, pagerank, indegree centrality, Katz’s index
2016  Common Search: host-level webgraph and pagerank [19, 20]
2017—now  host/domain-level webgraphs and rankings (harmonic centrality and pagerank) based on 3 monthly crawls
  - publicly released webgraph datasets
  - used to “steer” the crawler for the next three crawls
  - harmonic centrality more robust against link spam than page rank [21]
### Top-N .edu Domains Ranked by Harmonic Centrality [22]

<table>
<thead>
<tr>
<th>hc-rank</th>
<th>pagerank</th>
<th>Domain</th>
<th>hc-rank</th>
<th>pagerank</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>78</td>
<td>337</td>
<td>edu.stanford</td>
<td>297</td>
<td>795</td>
<td>edu.umich</td>
</tr>
<tr>
<td>96</td>
<td>302</td>
<td>edu.mit</td>
<td>321</td>
<td>964</td>
<td>edu.jhu</td>
</tr>
<tr>
<td>98</td>
<td>295</td>
<td>edu.harvard</td>
<td>322</td>
<td>940</td>
<td>edu.umn</td>
</tr>
<tr>
<td>136</td>
<td>510</td>
<td>edu.berkeley</td>
<td>332</td>
<td>1505</td>
<td>edu.indiana</td>
</tr>
<tr>
<td>177</td>
<td>506</td>
<td>edu.cornell</td>
<td>334</td>
<td>1079</td>
<td>edu.uchicago</td>
</tr>
<tr>
<td>180</td>
<td>523</td>
<td>edu.yale</td>
<td>344</td>
<td>1663</td>
<td>edu.gatech</td>
</tr>
<tr>
<td>184</td>
<td>747</td>
<td>edu.upenn</td>
<td>348</td>
<td>1042</td>
<td>edu.utexas</td>
</tr>
<tr>
<td>212</td>
<td>809</td>
<td>edu.washington</td>
<td>381</td>
<td>1121</td>
<td>edu.nyu</td>
</tr>
<tr>
<td>231</td>
<td>876</td>
<td>edu.asu</td>
<td>394</td>
<td>2736</td>
<td>edu.byu</td>
</tr>
<tr>
<td>253</td>
<td>875</td>
<td>edu.wisc</td>
<td>405</td>
<td>955</td>
<td>edu.si</td>
</tr>
<tr>
<td>268</td>
<td>1181</td>
<td>edu.umd</td>
<td>438</td>
<td>593</td>
<td>edu.cmu</td>
</tr>
<tr>
<td>271</td>
<td>1144</td>
<td>edu.purdue</td>
<td>445</td>
<td>1767</td>
<td>edu.tufts</td>
</tr>
<tr>
<td>286</td>
<td>1007</td>
<td>edu.princeton</td>
<td>449</td>
<td>1371</td>
<td>edu.duke</td>
</tr>
</tbody>
</table>
Domain-level harmonic centrality ranks are used

- to define a “budget” [23] for every domain how many URLs/pages are sampled or fetched
- to sample sitemaps or home pages for URL discovery (always for top-ranking domains, sometimes for lower ranks)
- as domain-level scores “projected” to the page-level by OPIC [24] or inlink counts

Per-domain limits (2022)

- top domains: 35 million URLs, 150k URLs per host, 500k subdomains
- long tail (below rank 64M or yet unseen): 1k URLs, 800 per host, 6 subdomains
- log distribution between top and tail
Data Collection – Web Crawler

Data Collection – Link Prioritization

Data Collection – Representativity, Geographical and Language Bias

- Are the Common Crawls Representative?
- New URLs and Domain Coverage
- Geographical Coverage
- Language Coverage
- Reasons why English Content is potentially overrepresented
- Fetch time by top-level domain
- Link Spam – Challenging the crawler
- Link Spam Detection i
- Link Spam Detection ii

Data – WARC and ARC Files

Data – Derivative Formats

Data – Usage

NLP Usage Examples

Summary
Are the Common Crawls Representative?

Aspects of representativity:

- breadth: coverage of unique domains (web sites)
- depth: per-site coverage
- regional coverage (top-level domains, content languages)
- amount of (near-)duplicates (both per crawl and over multiple crawl datasets)
New URLs and Domain Coverage

New URLs per Crawl (not observed in prior crawls)

Unique Domains per Crawl
Geographical Coverage

... by country-code top-level domain (percent of pages)
## Language Coverage

<table>
<thead>
<tr>
<th>Language</th>
<th>2018</th>
<th>2019</th>
<th>2020</th>
<th>2021</th>
<th>2022</th>
</tr>
</thead>
<tbody>
<tr>
<td>ces</td>
<td>1.033</td>
<td>1.037</td>
<td>1.051</td>
<td>1.060</td>
<td>1.031</td>
</tr>
<tr>
<td>deu</td>
<td>5.149</td>
<td>5.471</td>
<td>5.573</td>
<td>5.632</td>
<td>5.603</td>
</tr>
<tr>
<td>eng</td>
<td>43.958</td>
<td>43.835</td>
<td>43.200</td>
<td>44.981</td>
<td>46.639</td>
</tr>
<tr>
<td>fin</td>
<td>0.361</td>
<td>0.386</td>
<td>0.403</td>
<td>0.390</td>
<td>0.397</td>
</tr>
<tr>
<td>fra</td>
<td>4.530</td>
<td>4.560</td>
<td>4.527</td>
<td>4.456</td>
<td>4.500</td>
</tr>
<tr>
<td>jpn</td>
<td>5.469</td>
<td>4.808</td>
<td>4.784</td>
<td>4.665</td>
<td>4.685</td>
</tr>
<tr>
<td>mal</td>
<td>0.018</td>
<td>0.018</td>
<td>0.019</td>
<td>0.020</td>
<td>0.021</td>
</tr>
<tr>
<td>nno</td>
<td>0.013</td>
<td>0.019</td>
<td>0.019</td>
<td>0.018</td>
<td>0.016</td>
</tr>
<tr>
<td>nor</td>
<td>0.298</td>
<td>0.321</td>
<td>0.348</td>
<td>0.346</td>
<td>0.329</td>
</tr>
<tr>
<td>rus</td>
<td>9.274</td>
<td>7.406</td>
<td>7.113</td>
<td>7.067</td>
<td>5.897</td>
</tr>
<tr>
<td>swe</td>
<td>0.746</td>
<td>0.762</td>
<td>0.755</td>
<td>0.718</td>
<td>0.679</td>
</tr>
<tr>
<td>zho</td>
<td>4.978</td>
<td>6.763</td>
<td>6.914</td>
<td>5.067</td>
<td>4.733</td>
</tr>
</tbody>
</table>

- percentage of pages by language and year
- language identified by CLD2 since Aug 2018
- few languages shown, source [25]
Reasons why English Content is potentially overrepresented

Accept-Language HTTP header: `en-US, en; q=0.5`

- multi-lingual sites may show English content first
- or redirect to English (sub)site

Crawler is operated from data center located in the US (Northern Virginia)

- multi-lingual sites may show or redirect to language/region-specific site based on geo-located request IP address
- content from sites hosted nearby (given network topology) are favored because of shorter fetch times
<table>
<thead>
<tr>
<th>tld</th>
<th>ms/100kB</th>
<th>avg. page kB</th>
<th>ms/page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ca</td>
<td>596.2</td>
<td>148.0</td>
<td>882.1</td>
</tr>
<tr>
<td>us</td>
<td>651.2</td>
<td>137.5</td>
<td>895.1</td>
</tr>
<tr>
<td>co</td>
<td>664.2</td>
<td>146.3</td>
<td>971.9</td>
</tr>
<tr>
<td>dk</td>
<td>667.8</td>
<td>146.1</td>
<td>975.4</td>
</tr>
<tr>
<td>com</td>
<td>671.5</td>
<td>152.9</td>
<td>1026.7</td>
</tr>
<tr>
<td>ar</td>
<td>720.8</td>
<td>175.6</td>
<td>1265.6</td>
</tr>
<tr>
<td>ch</td>
<td>724.9</td>
<td>153.7</td>
<td>1114.1</td>
</tr>
<tr>
<td>gov</td>
<td>725.7</td>
<td>122.8</td>
<td>891.4</td>
</tr>
<tr>
<td>no</td>
<td>727.2</td>
<td>134.0</td>
<td>974.3</td>
</tr>
<tr>
<td>fi</td>
<td>754.0</td>
<td>131.1</td>
<td>988.1</td>
</tr>
<tr>
<td>ru</td>
<td>769.7</td>
<td>123.9</td>
<td>953.8</td>
</tr>
<tr>
<td>be</td>
<td>785.5</td>
<td>133.6</td>
<td>1049.1</td>
</tr>
<tr>
<td>org</td>
<td>793.5</td>
<td>113.5</td>
<td>900.8</td>
</tr>
<tr>
<td>edu</td>
<td>810.7</td>
<td>83.3</td>
<td>675.2</td>
</tr>
<tr>
<td>gr</td>
<td>818.3</td>
<td>165.0</td>
<td>1350.3</td>
</tr>
<tr>
<td>se</td>
<td>819.1</td>
<td>132.0</td>
<td>1080.8</td>
</tr>
<tr>
<td>ie</td>
<td>834.6</td>
<td>162.1</td>
<td>1352.9</td>
</tr>
<tr>
<td>ua</td>
<td>842.7</td>
<td>124.4</td>
<td>1048.5</td>
</tr>
<tr>
<td>at</td>
<td>845.2</td>
<td>132.4</td>
<td>1118.7</td>
</tr>
<tr>
<td>fr</td>
<td>849.0</td>
<td>131.0</td>
<td>1111.8</td>
</tr>
<tr>
<td>ro</td>
<td>854.7</td>
<td>134.3</td>
<td>1147.5</td>
</tr>
<tr>
<td>pl</td>
<td>864.3</td>
<td>123.6</td>
<td>1068.1</td>
</tr>
<tr>
<td>sk</td>
<td>870.6</td>
<td>135.4</td>
<td>1179.0</td>
</tr>
<tr>
<td>in</td>
<td>877.1</td>
<td>157.6</td>
<td>1381.9</td>
</tr>
<tr>
<td>de</td>
<td>891.5</td>
<td>125.0</td>
<td>1114.5</td>
</tr>
<tr>
<td>hu</td>
<td>892.6</td>
<td>134.0</td>
<td>1196.3</td>
</tr>
<tr>
<td>net</td>
<td>894.2</td>
<td>112.2</td>
<td>1003.0</td>
</tr>
<tr>
<td>pt</td>
<td>931.9</td>
<td>137.4</td>
<td>1280.7</td>
</tr>
<tr>
<td>cz</td>
<td>964.0</td>
<td>104.3</td>
<td>1005.0</td>
</tr>
<tr>
<td>nl</td>
<td>978.7</td>
<td>121.5</td>
<td>1189.3</td>
</tr>
<tr>
<td>es</td>
<td>979.3</td>
<td>130.5</td>
<td>1277.8</td>
</tr>
<tr>
<td>uk</td>
<td>980.9</td>
<td>91.3</td>
<td>895.7</td>
</tr>
<tr>
<td>cl</td>
<td>985.2</td>
<td>141.9</td>
<td>1398.2</td>
</tr>
<tr>
<td>eu</td>
<td>1023.4</td>
<td>122.8</td>
<td>1256.9</td>
</tr>
<tr>
<td>it</td>
<td>1068.1</td>
<td>130.9</td>
<td>1397.9</td>
</tr>
<tr>
<td>info</td>
<td>1107.0</td>
<td>96.5</td>
<td>1068.4</td>
</tr>
<tr>
<td>br</td>
<td>1121.1</td>
<td>112.6</td>
<td>1262.1</td>
</tr>
<tr>
<td>kr</td>
<td>1256.2</td>
<td>107.9</td>
<td>1355.7</td>
</tr>
<tr>
<td>jp</td>
<td>1356.4</td>
<td>91.8</td>
<td>1244.7</td>
</tr>
<tr>
<td>id</td>
<td>1538.7</td>
<td>112.9</td>
<td>1737.4</td>
</tr>
<tr>
<td>vn</td>
<td>1543.5</td>
<td>126.6</td>
<td>1954.0</td>
</tr>
<tr>
<td>ir</td>
<td>1652.8</td>
<td>115.9</td>
<td>1916.3</td>
</tr>
<tr>
<td>cn</td>
<td>1838.2</td>
<td>62.9</td>
<td>1156.5</td>
</tr>
</tbody>
</table>
• spam is part of the web, it’s ok if some is contained in the data
• October 2017: the crawler hit a spam cluster
  ▪ crawled: 56 million pages (1.5% of the crawl), 70,000 domains
  ▪ known from links: 320,000 domains, 2.5 billion subdomains
• highly branching spam clusters expensive for a crawler: every subdomain requires DNS look-up and robots.txt fetch/caching
• measures: set limit of crawled subdomains per domain and try to detect and block the worst link spam clusters
- spam clusters are volatile
- must detect spam with (almost) no training data
- need binary rule (is a spam domain or not)
- simple heuristics proved to work with little supervision based on imbalances between
  - centrality score
  - outgoing links
  - number of subdomains
  
  low-ranking domains with too many outlinks or subdomains are suspicious
- once some nodes of a spam cluster are identified, other nodes are easily found by looking for a strongly connected subcluster in the graph
Example based on the Jun/Jul/Sep 2021 domain-level graph, taking as spam indicator an exceptionally high product of harmonic centrality rank and number of known subdomains.

<table>
<thead>
<tr>
<th>sort</th>
<th>$\log_2(r \cdot n)$</th>
<th>hc rank $r$</th>
<th>$n$ subdomains</th>
<th>domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>44.93</td>
<td>33827380</td>
<td>993576</td>
<td>6suqmv2.site</td>
</tr>
<tr>
<td>2</td>
<td>44.34</td>
<td>50162956</td>
<td>445037</td>
<td>1st-muscle-guide.com</td>
</tr>
<tr>
<td>3</td>
<td>44.25</td>
<td>34012364</td>
<td>616681</td>
<td>wcpeoxt.icu</td>
</tr>
<tr>
<td>4</td>
<td>44.16</td>
<td>60683905</td>
<td>323917</td>
<td>ehime-di.com</td>
</tr>
<tr>
<td>5</td>
<td>44.09</td>
<td>36323509</td>
<td>515162</td>
<td>7ikoqnp.site</td>
</tr>
<tr>
<td>6</td>
<td>44.06</td>
<td>34195171</td>
<td>536038</td>
<td>m85g3vs.site</td>
</tr>
<tr>
<td>7</td>
<td>44.04</td>
<td>33824385</td>
<td>536460</td>
<td>5esg5j6.site</td>
</tr>
<tr>
<td>8</td>
<td>44.02</td>
<td>34925230</td>
<td>509545</td>
<td>mqv4s31.icu</td>
</tr>
<tr>
<td>9</td>
<td>43.90</td>
<td>33701024</td>
<td>487860</td>
<td>dcw7v3.xyz</td>
</tr>
<tr>
<td>10</td>
<td>43.81</td>
<td>35522472</td>
<td>433766</td>
<td>8s60fy.xyz</td>
</tr>
<tr>
<td>11</td>
<td>43.81</td>
<td>36357152</td>
<td>423051</td>
<td>76m30o.xyz</td>
</tr>
<tr>
<td>12</td>
<td>43.80</td>
<td>34202757</td>
<td>448281</td>
<td>x80u6n.xyz</td>
</tr>
<tr>
<td></td>
<td>2371148</td>
<td>26.89</td>
<td>24</td>
<td>5176495</td>
</tr>
<tr>
<td></td>
<td>2767418</td>
<td>26.67</td>
<td>18</td>
<td>5913686</td>
</tr>
</tbody>
</table>

...
Data Collection – Web Crawler
Data Collection – Link Prioritization
Data Collection – Representativity, Geographical and Language Bias

Data – WARC and ARC Files

The WARC format (Web ARChive)
The WARC format (example record)
The ARC format
The ARC format (example record)
Common Crawl WARC Specifics

Data – Derivative Formats

Data – Usage

NLP Usage Examples

Summary
The WARC format (Web ARChive)

- "freezes" the internet traffic between a client (web crawler or browser) and web servers at the HTTP protocol level
  - content payload
  - HTTP headers
  - connection metadata (datetime, IP address)
- WARC I/O modules for many programming languages [26]
- ISO standard since 2009 [27, 28]
- per-record gzipped: extract single records if offsets are known

```
```
The WARC format (example record)

WARC/1.0
WARC-Type: response
WARC-Date: 2022-12-09T06:54:33Z
Content-Length: 21810
WARC-IP-Address: 129.240.189.181
WARC-Target-URI: http://wiki.nlpl.eu/Home
WARC-Payload-Digest: sha1:3VWAF5J1DC46G5OYZGO6YI4ZVFTSDC45
WARC-Identified-Payload-Type: text/html

HTTP/1.1 200 OK
Date: Fri, 09 Dec 2022 06:54:32 GMT
Server: Apache/2.4.37 (Red Hat Enterprise Linux) SVN/1.10.2
X-Powered-By: PHP/7.2.24
Content-language: en
Last-Modified: Tue, 12 Jan 2021 18:42:52 GMT
X-Crawler-Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
Content-Length: 21223

<!DOCTYPE html>
<html class="client-nojs" lang="en" dir="ltr">
<head>
<meta charset="UTF-8"/>
<title>Home - Nordic Language Processing Laboratory</title>
<meta name="generator" content="MediaWiki 1.31.10"/>
<link rel="license" href="https://creativecommons.org/licenses/by/4.0/"/>
...

34
The ARC format

- until 2012 (legacy format)
- similar to WARC but
- capture metadata in single line

```
curl -s -r8067801-$(8067801+5288-1)) \
"https://data.commoncrawl.org/parse-output/segment/"\n"1346823846150/1346838136740_5172.arc.gz" \ 
| gzip -dc
```
The ARC format (example record)

http://www.commoncrawl.org/ 184.73.222.157 20120204064938 text/html 19468
HTTP/1.1 200 OK
Date:Sat, 04 Feb 2012 06:50:18 GMT
Server:Apache/2.2.17 (Ubuntu)
X-Powered-By:PHP/5.3.5-1ubuntu7
Last-Modified:Sat, 04 Feb 2012 06:50:18 GMT
Content-Type:text/html; charset=UTF-8
x-commoncrawl-DetectedCharset=UTF-8

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" dir="ltr" lang="en-US">
<head profile="http://gmpg.org/xfn/11">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>CommonCrawl | | CommonCrawl</title>
...

Specifcics of Common Crawl WARC and ARC collections (in difference to other web archivers)

- keep successful fetches (HTTP status 200) only
- separate subsets provided since autumn 2016 [29]
  - robots.txt
  - responses with HTTP status other than 200
    - 404 Not Found, 304 Not Modified, redirects, etc.
- removed HTTP content and transfer encoding (decompress, unchunk)
- page dependencies (images, CSS, etc.) not captured
Data Collection – Web Crawler

Data Collection – Link Prioritization

Data Collection – Representativity, Geographical and Language Bias

Data – WARC and ARC Files

Data – Derivative Formats

- WAT and WET
- URL Index

Data – Usage

NLP Usage Examples

Summary
WAT

- WARC and HTTP header fields
- HTML meta data
- links and attributes, eg. alt text

WET

- extracted plain text
- no markup or removal of boilerplate content (navigation, header, footer)

WAT and WET

- derivatives of the WARC file format
  - WARC headers
  - payload: JSON or text
- legacy code base [30, 31]
- planned replacement by columnar data format in the long term
- captures are (randomly) distributed over WARC files
- index to look up URL to get location of WARC capture plus metadata

**index.commoncrawl.org**
- format: ZipNum Sharded CDX [32]
- main crawls 2008 – now

**index in columnar format (Parquet) [33, 34]**
- SQL queries and aggregations using big data tools (Spark, Hive, Presto/Trino/Athena)
- main crawls 2013 – now
Data Collection – Web Crawler

Data Collection – Link Prioritization

Data Collection – Representativity, Geographical and Language Bias

Data – WARC and ARC Files

Data – Derivative Formats

Data – Usage

Data Size and Usage By Format

Data usage by capture time and format

NLP Usage Examples

Summary
Data Size and Usage By Format

- percentage of occupied storage and request volume by format in 2022
- "reads": ratio request / storage volume (≈ times volume of provided data is read)
- users prefer text extracts and indexes (processed, condensed, small)
- WAT metadata extracts less popular despite the smaller size (compared to WARC)
- columnar Parquet index heavily used

<table>
<thead>
<tr>
<th>format</th>
<th>% data</th>
<th>% requ.</th>
<th>reads</th>
</tr>
</thead>
<tbody>
<tr>
<td>WARC/ARC</td>
<td>69.0</td>
<td>63.0</td>
<td>23</td>
</tr>
<tr>
<td>WAT</td>
<td>21.3</td>
<td>8.3</td>
<td>10</td>
</tr>
<tr>
<td>WET</td>
<td>9.1</td>
<td>16.8</td>
<td>46</td>
</tr>
<tr>
<td>CDX Index</td>
<td>0.4</td>
<td>0.7</td>
<td>24</td>
</tr>
<tr>
<td>Parq. Index</td>
<td>0.3</td>
<td>11.6</td>
<td>1045</td>
</tr>
</tbody>
</table>
Data usage by capture time and format

- Users generally prefer to use recently harvested web data
- Data captured years ago: text extracts and indexes more popular
- Challenges (given our mission is to enable web data usage)
  - Better tooling and documentation for the ARC format (2008 – 2012)
  - Replacement for the WAT format (metadata and hyperlinks)
NLP Usage Examples

- utilize existing examples, tools, libraries [35, 36]
- select the right data set and format
- is there already a third-party dataset easier to use?
- ask for help https://groups.google.com/g/common-crawl
Requirements and recommendations

- use a WARC parser library
  - WAT or WET are WARC derivatives
- processing of files easy to parallelize
  - Hadop Map-Reduce, Spark, or any other framework
  - which one to choose determined by how data is further consumed (complex reducer or filter pipeline, etc.)
example by Philippe Suter [37]

- **z-index** – CSS property to set the z-order of a positioned element in the web browser
- elements with a higher z-index are “on top” of the overlay stack, not hidden by elements with lower z-index
- which z-index values are chosen by web developers?

```html
<script>
(function () {
    window.loadAndOpenZendeskChat();
    // var btnHtml = '<iframe id="zdbutton" ... title="Support Chat" tabindex="0"
    // style="width: 110px; height: 50px; padding: 0px; margin: 10px 20px;
    // position: fixed; bottom: 0px; overflow: visible; opacity: 1; border: 0px none; z-index: 999998; ...">
    // lxml
    // ..."/>
})( ) ;
</script>
```
Bulk Processing: z-index example

z-index usage in a sample of web pages

- define a regular expression to extract z-index values
  ```python
  re.compile(b'z-index *: *(-?[0-9]+|auto|inherit|initial|unset)')
  ```
- find the right example to build upon [38, 39]
- implement the Spark job and run it
  ```bash
  spark-submit z_index_count.py --output_format json .../path/to/input-warc.paths zindex_count
  ```
import re
from collections import Counter
from sparkcc import CCSparkJob

class ZIndexCountJob(CCSparkJob):
    """Count z-index values in Common Crawl WARC files,
    cf. https://psuter.net/2019/07/07/z-index"""
    name = "ZIndexCount"

    zindex_pattern = re.compile(b'z-index *: *(-?[0-9]+|auto|inherit|initial|unset)')

    def process_record(self, record):
        if record.rec_type != 'response':
            # skip over WARC request or metadata records
            return
        if not self.is_html(record):
            # skip non-HTML or unknown content types
            return
        data = record.content_stream().read()
        counts = Counter(ZIndexCountJob.zindex_pattern.findall(data))
        for val, count in counts.items():
            yield val.decode('ascii'), count

if __name__ == '__main__':
    job = ZIndexCountJob()
    job.run()
Figure 1 – Most commonly used *z-index* values. The y axis shows the relative frequency.
Bulk Processing: Spark job explained

... behind the scenes

- Spark reads the input list of WARC paths and distributes it to workers
- the mapper function (called by the worker, implemented in CCSparkJob) opens a WARC file
- the WARC input stream is passed to warcio (WARC parser module)
- iterating over WARC records, the method `process_record` is called
- output tuples are grouped by key and passed to the default reducer (sum values)
the columnar index [33, 40, 34] includes

- URL and parts (host name, registered domain, top-level domain, path, query)
- capture metadata (fetch time, size, WARC record location)
- content metadata (MIME type, charset, content languages detected by CLD2)
- see the table schema [41, 42] for details

the following example queries were run with Amazon Athena [43] engine v3
SELECT COUNT(*) AS n_captures, COUNT(DISTINCT url_host_registered_domain) AS n_domains, url_host_tld
FROM "ccindex"."ccindex"
WHERE crawl = 'CC-MAIN-2022-49'
AND subset = 'warc'
-- primary language: Malayalam
AND content_languages LIKE 'mal%'
GROUP BY url_host_tld
ORDER BY n_captures DESC;
WITH tmp AS (  
  SELECT COUNT(*) AS num_pages,
    regexp_extract(content_languages, '^[a-z]{3}$') AS primary_content_language,
    url_host_name,
    SUM(COUNT(*)) OVER (PARTITION BY regexp_extract(content_languages, '^[a-z]{3}$')) AS total_pages_lang,
    SUM(COUNT(*)) OVER (PARTITION BY url_host_name) AS total_pages_host,  
    array_agg(regexp_extract(content_languages, '^[a-z]{3}$')) OVER (PARTITION BY url_host_name) AS host_primary_content_languages
  FROM ccindex.ccindex
  WHERE crawl = 'CC-MAIN-2022-49'
  AND subset = 'warc'
  GROUP BY regexp_extract(content_languages, '^[a-z]{3}$'),
    url_host_tld,
    url_host_name)

SELECT num_pages,
  url_host_name,
  (100.0*num_pages/total_pages_lang) AS perc_of_lang,
  total_pages_host,
  (100.0*num_pages/total_pages_host) AS perc_of_host,
  host_primary_content_languages
FROM tmp
WHERE primary_content_language = 'mal'
AND num_pages >= 5
AND (100.0*num_pages/total_pages_host) >= 5.0
ORDER BY num_pages DESC;
see also `site-discovery-by-language.sql`

next slide: sample out of 2 478 results
## Exploration: examples of Malayalam sites

<table>
<thead>
<tr>
<th>pages</th>
<th>host</th>
<th>%lang</th>
<th>phost</th>
<th>%phost</th>
<th>primary languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>29856</td>
<td>ml.wikipedia.org</td>
<td>4.030</td>
<td>30161</td>
<td>98.99</td>
<td>[lat, mal, eng]</td>
</tr>
<tr>
<td>17873</td>
<td>malayalam.news18.com</td>
<td>2.413</td>
<td>17875</td>
<td>99.99</td>
<td>[eng, mal]</td>
</tr>
<tr>
<td>14528</td>
<td>malayalam.indianexpress.com</td>
<td>1.961</td>
<td>14528</td>
<td>100.00</td>
<td>[mal]</td>
</tr>
<tr>
<td>8169</td>
<td>fanport.in</td>
<td>1.103</td>
<td>8762</td>
<td>93.23</td>
<td>[mal, null, eng]</td>
</tr>
<tr>
<td>1835</td>
<td>celebrity.astrosage.com</td>
<td>0.248</td>
<td>13192</td>
<td>13.91</td>
<td>[ben, eng, mar, hin, guj, tel, mal]</td>
</tr>
<tr>
<td>969</td>
<td>onlinepeeps.co</td>
<td>0.131</td>
<td>1001</td>
<td>96.80</td>
<td>[eng, mal]</td>
</tr>
<tr>
<td>225</td>
<td>sapnageorge.com</td>
<td>0.030</td>
<td>364</td>
<td>61.81</td>
<td>[eng, mal]</td>
</tr>
<tr>
<td>207</td>
<td>kambistories.co***</td>
<td>0.028</td>
<td>207</td>
<td>100.00</td>
<td>[mal]</td>
</tr>
<tr>
<td>136</td>
<td><a href="http://www.mahzooz.ae">www.mahzooz.ae</a></td>
<td>0.018</td>
<td>672</td>
<td>20.24</td>
<td>[ara, urd, mal, hin, eng]</td>
</tr>
<tr>
<td>30</td>
<td>myday.code.blog</td>
<td>0.004</td>
<td>30</td>
<td>100.00</td>
<td>[mal]</td>
</tr>
<tr>
<td>18</td>
<td><a href="http://www.myjar.app">www.myjar.app</a></td>
<td>0.002</td>
<td>287</td>
<td>6.27</td>
<td>[lat, hin, mar, mal, tel, eng, kan]</td>
</tr>
<tr>
<td>12</td>
<td>malayalambible.app</td>
<td>0.002</td>
<td>15</td>
<td>80.00</td>
<td>[mal, eng]</td>
</tr>
<tr>
<td>11</td>
<td><a href="http://www.malayalambible.app">www.malayalambible.app</a></td>
<td>0.001</td>
<td>16</td>
<td>68.75</td>
<td>[eng, mal]</td>
</tr>
</tbody>
</table>

*** adult content
The Vertical Use Case

- URL index holds WARC record locations
  - WARC file path
  - record offset and length (response record)
- fetch a given WARC record by sending a HTTP range request using the offsets
- any query to the (columnar) URL index to select records
  - by content language, top-level domain, MIME type, keyword in URL path, ...
  - for a given list of host/domain names or URLs
    use SQL JOIN for any larger list (see [44])
- how to proceed with the returned gzipped WARC record(s)?
  - process the WARC records on-the-fly (see [34, 39] or [45])
  - concatenate compressed records into WARC file
example by Athul Jayson [46, 47]

- get WARC record locations
  
  ```sql
  SELECT url, 
         warc_filename, 
         warc_record_offset, 
         warc_record_length
  FROM "ccindex"."ccindex"
  WHERE crawl = 'CC-MAIN-2020-05'
       AND subset = 'warc'
       AND content_languages LIKE '%mal%'
  ```

- (distributed) fetching of WARC records and export into WARC files
  - Spark job defined in [47, 40]
  - run on AWS, us-east-1 where CC data is stored (low network latency, cf. [48])

- local processing of WARC files: parse HTML and extract Malayalam text [47]
Summary

data collection

- simple and polite crawler
  - substantial parts of the web are missing
- sample crawls, not exhaustive (not every domain, not every page from a site)
- not free of collection bias

data usage

- have a look at example projects
- take time to select the best data format and tool
- stay in touch
Questions?


References


