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

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

	crawler	seeds / link prioritization
2008–2012	in-house	pagerank
2013–2016	Nutch	blekko seed donation
2017– now	Nutch	harmonic centrality

Web Crawler 2008 – 2012

- 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

Seed Donations 2013 – 2018

2013–2015 web search engine blekko [5] donates ranking and metadata of 140 million web sites and 22 billion pages

> *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

Web Crawler 2013 – now

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. **dedup**licate documents by signature
7. **index** document content, meta data, and anchor texts
8. (data exports)

Nutch at Common Crawl 2013 – 2016

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

Nutch at Common Crawl 2017 – now

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

Nutch at Common Crawl – Good to know

- 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

Nutch at Common Crawl – Hadoop and hardware

- 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)

Nutch at Common Crawl – Fetch list layout

- 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)

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

Link Prioritization – Web Graphs and rankings based on Common Crawl

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]

Link Prioritization – Graph-based ranking example

top-N .edu domains ranked by harmonic centrality [22]

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

Prioritization – A Deeper Look into the Current Implementation i

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)

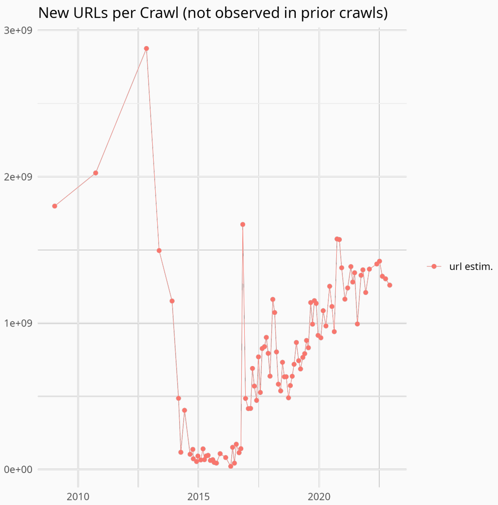
- top domains: 35 million URLs, 150k URLs per host, 500k subdomains
- long tail (below rank 64M or yet unseen): 1k URLs, 800 per host, 6 subdomains
- log distribution between top and tail

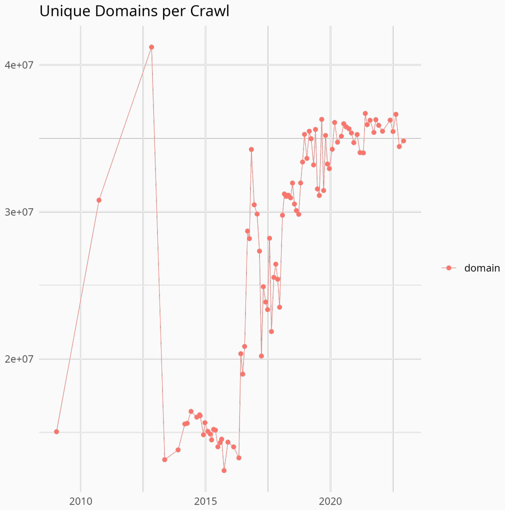
Are the Common Crawls Representative?

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)

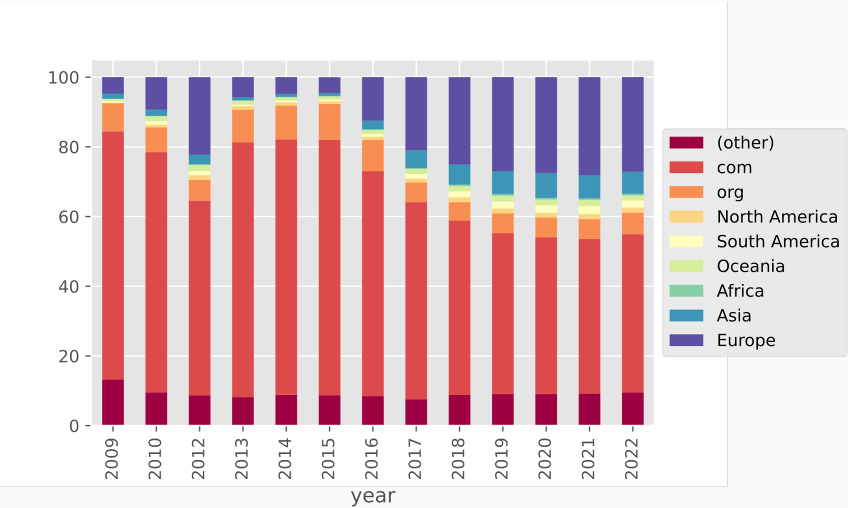
New URLs and Domain Coverage





Geographical Coverage

… by country-code top-level domain (percent of pages)



Language Coverage

%	2018	2019	2020	2021	2022
ces	1.033	1.037	1.051	1.060	1.031
deu	5.149	5.471	5.573	5.632	5.603
eng	43.958	43.835	43.200	44.981	46.639
fin	0.361	0.386	0.403	0.390	0.397
fra	4.530	4.560	4.527	4.456	4.500
jpn	5.469	4.808	4.784	4.665	4.685
mal	0.018	0.018	0.019	0.020	0.021
nno	0.013	0.019	0.019	0.018	0.016
nor	0.298	0.321	0.348	0.346	0.329
rus	9.274	7.406	7.113	7.067	5.897
spa	4.179	4.160	4.248	4.330	4.356
swe	0.746	0.762	0.755	0.718	0.679
zho	4.978	6.763	6.914	5.067	4.733

- 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

Fetch time by top-level domain

tld	ms/100kiB	avg. page kiB	ms/page
ca	596.2	148.0	882.1
us	651.2	137.5	895.1
co	664.2	146.3	971.9
dk	667.8	146.1	975.4
com	671.5	152.9	1026.7
ar	720.8	175.6	1265.6
ch	724.9	153.7	1114.1
gov	725.7	122.8	891.4
no	727.2	134.0	974.3
fi	754.0	131.1	988.1
ru	769.7	123.9	953.8
be	785.5	133.6	1049.1
org	793.5	113.5	900.8
edu	810.7	83.3	675.2
gr	818.3	165.0	1350.3
se	819.1	132.0	1080.8
ie	834.6	162.1	1352.9
ua	842.7	124.4	1048.5
at	845.2	132.4	1118.7
fr	849.0	131.0	1111.8
ro	854.7	134.3	1147.5
pl	864.3	123.6	1068.1
sk	870.6	135.4	1179.0
in	877.1	157.6	1381.9
de	891.5	125.0	1114.5
hu	892.6	134.0	1196.3
net	894.2	112.2	1003.0
pt	931.9	137.4	1280.7
cz	964.0	104.3	1005.0
nl	978.7	121.5	1189.3
es	979.3	130.5	1277.8
uk	980.9	91.3	895.7
cl	985.2	141.9	1398.2
eu	1023.4	122.8	1256.9
it	1068.1	130.9	1397.9
info	1107.0	96.5	1068.4
br	1121.1	112.6	1262.1
kr	1256.2	107.9	1355.7
jp	1356.4	91.8	1244.7
id	1538.7	112.9	1737.4
vn	1543.5	126.6	1954.0
ir	1652.8	115.9	1916.3
cn	1838.2	62.9	1156.5

Link Spam – Challenging the crawler

- 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

Link Spam Detection i

- 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

Link Spam Detection ii

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

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

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

```
curl -s -r56202708-$((56202708+7445-1)) \
"https://data.commoncrawl.org/crawl-data/CC-MAIN-2022-49/segments/"\
"1669446711390.55/warc/CC-MAIN-20221209043931-20221209073931-00615.warc.gz" \
  | gzip -dc
```

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: sha1:3VWAF5JIDC46G5OYZGO6YI4ZVFTSDC45
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/"/>
...
```

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/"\
"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-transiti
<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>
...
```

Common Crawl WARC Specifics

Specifics 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 and WET

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

URL Index

- 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

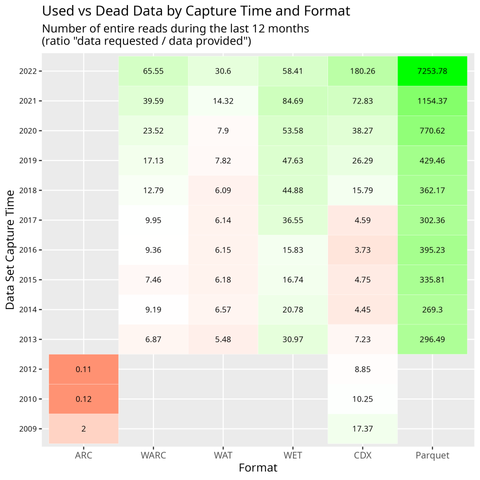
Data Size and Usage By Format

- 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

format	% data	% requ.	reads
WARC/ARC	69.0	63.0	23
WAT	21.3	8.3	10
WET	9.1	16.8	46
CDX Index	0.4	0.7	24
Parq. Index	0.3	11.6	1045

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)



- 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

Bulk Processing WARC, WAT or WET

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?

```
<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; ..."></iframe>';
   }());
</script>
```

Bulk Processing: z-index example

z-index usage in a sample of web pages

- define a regular expression to extract z-index values

```
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

```
spark-submit z_index_count.py --output_format json .../path/to/input-warc.paths zindex_count
```

Bulk Processing: Spark job definition

```python
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"

    # match z-index values on binary HTML data (in embedded CSS)
    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()
```

Bulk Processing: z-index value counts

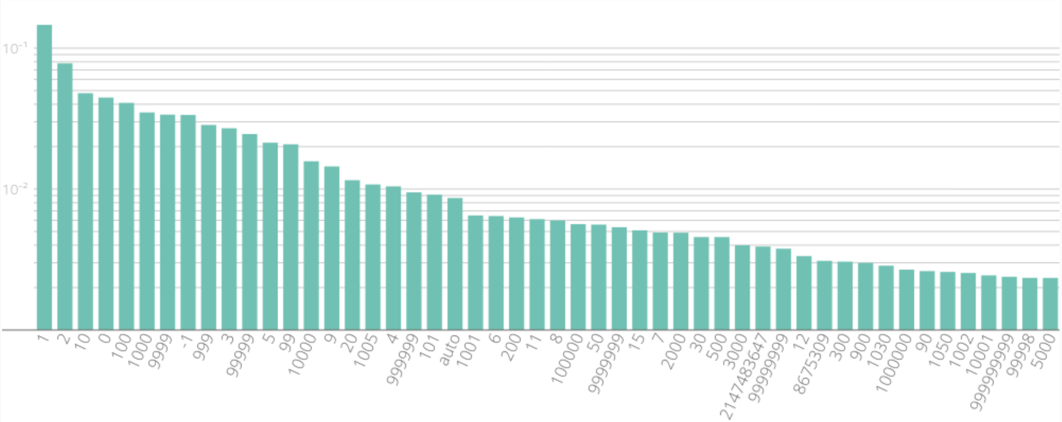


Figure 1 – Most commonly used `z-index` values. The y axis shows the relative frequency.

Bulk Processing: Spark job explained

… behinde 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)

Exploration on URL and Metadata Index

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

Exploration: top-level domains hosting Malayalam content

```sql
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;
```

509602	3243	com
111051	805	in
82183	427	org
8592	133	net
6909	13	news
5098	5	tw
3466	41	info
1746	10	co
1717	4	media
1265	2	cyou
1007	8	blog
843	10	online
798	9	live
730	16	me
594	9	app
499	6	xyz
434	15	tv
360	3	ml
342	5	ae
333	10	de
283	3	xn--rvc1e0am3e

Exploration: discovery of sites hosting Malayalam content

```
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;
```

Exploration: discovery of sites hosting Malayalam content

- see also site-discovery-by-language.sql
- next slide: sample out of 2 478 results

Exploration: examples of Malayalam sites

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

The Vertical Use Case: Malayalam Text Corpus

example by Athul Jayson [46, 47]

- get WARC record locations

```
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?

[1] Amazon Web Services. *Open Data Sponsorship Program*. https://aws.amazon.com/opendata/open-data-sponsorship-program/.

[2] Amazon Web Services. *Registry of Open Data on AWS*. https://registry.opendata.aws/.

[3] Ahad Rana. *Common Crawl – Building an open web-scale crawl using Hadoop*. 2010. https://www.slideshare.net/hadoopusergroup/common-crawlpresentation.

[4] *The Common Crawl Crawler Engine and Related MapReduce code (2008-2012)*. https://github.com/commoncrawl/commoncrawl-crawler.

[5] *blekko*. https://en.wikipedia.org/wiki/Blekko.

[6] *blekko donates search data to Common Crawl*. 2012. https://commoncrawl.org/2012/12/blekko-donates-search-data-to-common-crawl/.

[7] *Apache Nutch*. https://nutch.apache.org/.

[8] *Jordan Mendelson 2014: Common Crawl's Move to Nutch*. https://commoncrawl.org/2014/02/common-crawl-move-to-nutch/.

[9] Andrzej Białecki. “Nutch Search Engine”. In: *Hadoop: The Definitive Guide*. Ed. by Tom White. O’Reilly, 2012, pp. 565–579.

[10] *Common Crawl Fork of Apache Nutch*. https://github.com/commoncrawl/nutch.

[11] *sitemaps.org*. https://www.sitemaps.org/protocol.html.

[12] *Oracle Data Cloud Crawler*. https://www.oracle.com/corporate/acquisitions/grapeshot/crawler.html.

[13] *News Dataset Available*. 2016. https://commoncrawl.org/2016/10/news-dataset-available/.

[14] *News-Crawl*. https://github.com/commoncrawl/news-crawl.

[15] *StormCrawler*. https://stormcrawler.net/.

[16] *Web Data Commons - Hyperlink Graphs*. 2013. https://webdatacommons.org/hyperlinkgraph/index.html.

[17] Robert Meusel et al. “The Graph Structure in the Web – Analyzed on Different Aggregation Levels”. In: *The Journal of Web Science* 1.1 (2015), pp. 33–47. DOI: http://dx.doi.org/10.1561/106.00000003. https://pdfs.semanticscholar.org/b5d5/88298e6845b4bfd40ea779ce21e628239ef3.pdf.

[18] *The Common Crawl WWW Ranking*. http://wwwranking.webdatacommons.org/.

[19] *Common Search: Our first public datasets: Host-level WebGraph and PageRank.* https://web.archive.org/web/20170729110709/https://about.commonsearch.org/2016/07/our-first-public-datasets-host-level-webgraph-and-pagerank/.

[20] https://github.com/commonsearch/cosr-back/blob/master/spark/jobs/pagerank.py.

[21] Paolo Boldi. *A modern view of centrality measures.* 2013. https://events.yandex.ru/events/science-seminars/boldi-23sep.

[22] *Host- and Domain-Level Web Graphs May, June/July and August 2022.* https://commoncrawl.org/2022/09/host-and-domain-level-web-graphs-may-jun-aug-2022/.

[23] Hsin-Tsang Lee et al. “IRLbot: Scaling to 6 Billion Pages and Beyond”. In: *ACM Trans. Web* 3.3 (July 2009). ISSN: 1559-1131. DOI: 10.1145/1541822.1541823. https://doi.org/10.1145/1541822.1541823.

[24] Serge Abiteboul, Mihai Preda, and Gregory Cobena. “Adaptive on-line page importance computation”. In: (2003). https://dx.doi.org/10.1145/775152.775192.

[25] *Statistics of Common Crawl Monthly Archives.* https://commoncrawl.github.io/cc-crawl-statistics/.

[26] *Awesome Web Archiving*.

[27] Wikipedia contributors. *Web ARChive — Wikipedia, The Free Encyclopedia*. 2021. https://en.wikipedia.org/wiki/Web_ARChive.

[28] *The WARC Format 1.1*. https://iipc.github.io/warc-specifications/specifications/warc-format/warc-1.1/.

[29] *Data Sets Containing Robots.txt Files and Non-200 Responses*. https://commoncrawl.org/2016/09/robotstxt-and-404-redirect-data-sets/.

[30] *IIPC Web Archive Commons (Common Crawl Fork)*. https://github.com/commoncrawl/ia-web-commons.

[31] *IIPC Web Archive Commons*. https://github.com/iipc/webarchive-commons.

[32] *PyWB: CDX Index Format – ZipNum Sharded CDX*. https://github.com/webrecorder/pywb/wiki/CDX-Index-Format#zipnum-sharded-cdx.

[33] *Index to WARC Files and URLs in Columnar Format*. 2018. https://commoncrawl.org/2018/03/index-to-warc-files-and-urls-in-columnar-format/.

[34] Sebastian Nagel. *Accessing WARC files via SQL*. Poster at IIPC Web Archiving Conference, 6–7 June 2019, Zagreb, Croatia. 2019. https://digital.library.unt.edu/ark:/67531/metadc1608961/.

[35] *Examples using Common Crawl Data*. https://commoncrawl.org/the-data/examples/.

[36] *So you're ready to get started*. https://commoncrawl.org/the-data/get-started/.

[37] Philippe Suter. *Index fun*. 2019. https://psuter.net/2019/07/07/z-index.

[38] Mark Litwintschik. *Analysing Petabytes of Websites*. 2017. https://tech.marksblogg.com/petabytes-of-website-data-spark-emr.html.

[39] *Common Crawl PySpark Examples*. https://github.com/commoncrawl/cc-pyspark.

[40] *Common Crawl Index Table*. https://github.com/commoncrawl/cc-index-table.

[41] *Common Crawl Index Table (Data)*. https://data.commoncrawl.org/cc-index/table/cc-main/index.html.

[42] *Common Crawl Index Table, table schema flat*. https://github.com/commoncrawl/cc-index-table/blob/main/src/main/resources/schema/cc-index-schema-flat.json.

[43] *Interactive SQL - Serverless Query Service - Amazon Athena - AWS*. https://aws.amazon.com/athena/.

[44] *Table Joins to Look Up Large Lists of URLs in the Common Crawl*. https://github.com/commoncrawl/cc-notebooks/blob/main/cc-index-table/bulk-url-lookups-by-table-joins.ipynb.

[45] Jader Dias. *One click to download all the web pages you may want*. 2022. https://medium.com/@jaderd/one-click-to-download-exactly-the-web-pages-you-may-want-no-matter-how-many-they-are-d4834265a0a3.

[46] Athul Jayson. *Extracting Data from Common Crawl, QBurst Blog*. 2020. https://blog.qburst.com/2020/07/extracting-data-from-common-crawl-dataset/.

[47] *Common Crawl - Malayalam*. https://github.com/qburst/common-crawl-malayalam.

[48] *S3 Throughput: Scans vs Indexes*. 2019. https://code402.com/blog/s3-scans-vs-index/.