

Improved methodology for longitudinal Web analytics using Common Crawl

Henry S. Thompson
 The University of Edinburgh
 Edinburgh, United Kingdom

ABSTRACT

Common Crawl is a multi-petabyte longitudinal dataset containing over 100 billion web pages which is widely used as a source of language data for sequence model training and in web science research. Each of its constituent archives is on the order of 75TB in size. Using it for research, particularly longitudinal studies, which necessarily involve multiple archives, is therefore very expensive in terms of compute time and storage space and/or web bandwidth. Two new methods for mitigating this problem are presented here, based on exploiting and extending the much smaller (<200 gigabytes (GB) compressed) *index* which is available for each archive. By adding Last-Modified timestamps to the index we enable longitudinal exploration using only a single archive. By comparing the distribution of index features for each of the 100 segments into which archive is divided with their distribution over the whole archive, we have identified the least and most representative segments for a number of recent archives. Using this allows the segment(s) that are most representative of an archive to be used as proxies for the whole. We illustrate this approach in an analysis of changes in URI length over time, leading to an unanticipated insight into the how the creation of Web pages has changed over time.

CCS CONCEPTS

- **Information systems** → *Data extraction and integration; Digital libraries and archives; Web crawling*.

KEYWORDS

Common Crawl, Web crawling, Web archives

ACM Reference Format:

Henry S. Thompson. 2024. Improved methodology for longitudinal Web analytics using Common Crawl. In *ACM Web Science Conference (Websci '24), May 21–24, 2024, Stuttgart, Germany*. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3614419.3644018>

1 INTRODUCTION

Common Crawl [9] is a very-large-scale corpus containing petabytes of data from more than 100 archives. It contains over 100 billion web pages, more than 99% of which are HTML formatted, collected

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Websci '24, May 21–24, 2024, Stuttgart, Germany

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM. ACM ISBN 979-8-4007-0334-8/24/05...\$15.00
<https://doi.org/10.1145/3614419.3644018>

since 2008. It is hosted online free of charge courtesy of Amazon Web Services' Open Data Sponsorship program [1].

Existing Web Analytics literature focuses on small subsets of the Web, or only examines the distribution of Top-Level Domains (TLDs). Although Common Crawl offers a large and freely available source of data, it has rarely been used as a basis for studying the evolution of the Web.

One obvious reason for this is the considerable cost, both in compute time, web bandwidth and storage that is involved in processing multiple archives. The average size of a compressed archive has grown from 50 terabytes (TB) in 2019 to 100TB in 2023.

In this work we explore how to take advantage of two aspects of Common Crawl archives to mitigate this problem:

- (1) Each archive is packaged in 100 randomised subsets called *segments* of approximately equal size;
- (2) Each archive since 2013 comes with a sharded index [21], which provides descriptive metadata for every retrieved item therein.

The index for a given archive is available for separate download and is less than 200GB in size. It is thus possible to explore the evolution of a number of important aspects of Web architecture which are manifest in the index with much less computational cost compared to that required for working with the corresponding complete archives themselves.

Chen [6] suggested that *if* one could guarantee that "each segment [of a Common Crawl archive] has similar distribution [to the whole]", *then* processing only a single segment per year would provide comparable results to those for the whole, at 1% of the cost. We take up that suggestion here by using the index to measure the extent to which each segment is representative of the whole archive, we can use the segment(s) that are most representative of an archive as proxies for the whole, again reducing the computational cost involved even when the properties of interest are *not* available from the index.

In what follows we report on a two-part study. The first explores the use of index features from four archives from August/September of 2019, 2020, 2021 and 2023 to determine how we can best measure segment representativeness in order to chose proxies. The second uses this approach to evaluate the hypothesis that the length of URIs is growing, and that this growth is due primarily to a growth in the query URI component. We conclude with some surprising evidence for the ongoing change in the way web pages are created, from human authoring to automatic generation.

2 MATERIALS

"The Common Crawl corpus contains petabytes of data collected since 2008. It contains raw web page data, extracted metadata and text extractions.

Table 1: Scale of archives used

Archive ID	WARC files	WARC Size (compressed)
CC-MAIN-2019-35	56,000	54TB
CC-MAIN-2029-34	60,000	49TB
CC-MAIN-2021-31	72,000	75TB
CC-MAIN-2023-40	90,000	98TB

Table 2: Size of archive components (in millions of retrievals)

	2019-35	2020-34	2021-31	2023-40
WARC	2,955	2,450	3,165	3,445
non-200	549	520	465	553
robots.txt	114	108	86	90
Total	3,618	3,078	3,716	4,089

The Common Crawl dataset lives on Amazon S3 as part of the Amazon Web Services' Open Data Sponsorships program. You can download the files entirely free using HTTP(S) or S3.” [24]

Each Common Crawl archive has six main components [8]:

- (1) Data files
 - (a) Successful retrievals
 - (i) **WARC** files: Complete HTTP request and response; archive metadata
 - (ii) **WAT** files: Response metadata only
 - (iii) **WET** files: Text content (HTML responses only)
 - (b) Unsuccessful retrievals
 - (i) **robots.txt** files: HTTP headers and message body
 - (ii) **non-200 responses** files: HTTP headers and message body for e.g. not found, server fault, redirect
- (2) **URI index** files: 300 shards, alphabetical by URI inverse hostname, plus binary-searchable master index

All the constituent files are compressed, with the exception of the URI master index.

The data files are divided into 100 segments, numbered from 0 to 99, with equal numbers of files, but not necessarily of pages¹.

The details of the four Common Crawl archives we used are given in Tables 1 and 2. The two-digit number at the end of the Archive ID gives the week in which the archive was carried out. Each gzipped WARC-format [19] file in these archives contains around 50,000 records of successful HTTP(S) request-response exchanges. The CDX-format [20] index of request URIs provides file-ids, offset and length for all records in the WARC, robots.txt and non-200 response components (details below).

2.1 URL index

The index format is relatively simple: It consists of a compressed plaintext index (with one line for each

¹Strictly speaking the contents of archives are not pages, but records of HTTP requests and responses. For successful requests the response body is a (text/html or application/pdf or ...) representation of one or more pages, and we'll use *page* below in this informal way.

entry) compressed into gzipped chunks, and a [master] index of the compressed chunks. This index is often called the ‘ZipNum’ CDX format [20] and it is the same format that is used by the Wayback Machine at the Internet Archive.” [21]

Taken together, the two parts of the URI index make it possible to access a single request-response record, either from a local copy of the WARC component² or by via an HTTP request.

An abbreviated example is the best way to explain how this works. Suppose we want to see if the XML specification [23], whose URI is <https://www.w3.org/TR/xml/>, is in the August 2021 archive, for which we have the index.

Both the master index (`cluster.idx`) and the individual primary index files are keyed with a *urlkey*, which is a version of a URI, canonicalised using a version of the Internet Archive's *Sort-friendly URI Reordering Transform* [3]³:

- Remove http(s)://;
- replace A-Z with a-z throughout;
- if the authority component begins with www., remove it;
- reverse the order of the authority component, replace any periods with commas and insert a right-parenthesis at the end;
- if the path ends with a slash, remove it.

For our example, this would turn <https://www.w3.org/TR/xml/> into `org,w3,tr/xml`.

The lines of the primary and master index files consist of space-separated fields, the first of which is a urlkey, and the lines are in alphabetical order based on that field.

There are 300 primary index files, and each file is compressed into blocks of 3000 lines. The master index file contains one line for each of those compressed blocks, keyed by the urlkey for the first line in the block. The remaining fields in the master index give the primary index file name and the offset and length of the block in that file.

This provides for very efficient search for a URI, as follows:

- (1) Convert to a urlkey;
- (2) binary search in the master index for the last line whose key is less than or equal to that urlkey, and note the primary index file name, offset and length therein;
- (3) extract the block of the given length at the given offset from the gzipped version of the given primary index file and unzip it;
- (4) binary search therein for the last line whose key is less than or equal to your urlkey: if equal, you win, if not, you lose.

Note that this depends on an important property of the gzip compression format [12], which is used by Common Crawl for compression: a single compressed file may consist of the concatenation of separately compressed files. Not only the primary index files, but also the WARC files themselves exploit this. For each successful retrieval there is a separate gzipped block containing the three WARC-format records for HTTP request, HTTP response and archive metadata. This allows individual entries in the primary

²As noted earlier, the index actually covers the robots.txt and non-200 responses components as well, but the work reported here makes no use of that.

³Actual implementations not only do more than what is described here, they also differ with respect to the how they handle various corner cases, *caveat emptor*.

index file to reference individual retrievals by file ID, offset and length in a gzipped WARC file.

Although there are only on the order of 3×10^9 successful retrievals in recent archives, because the index also covers the **robots.txt** and **non-200** retrievals, there are around 3.6×10^9 index entries in total, split into 300 primary index files. Each individual index file thus has on the order of 1.2×10^7 lines. This in turn means that, since the compressed blocks in each file contain 3,000 lines, a single primary index file is split into about 4,000 blocks and the master index, with one line per block, has about 1.2 million lines.

Looking up a given URI thus takes around 21 tests in the master index (step 2 above) and 12 tests in the resulting primary index block (step 4) and only requires unzipping one 3,000-line block.

Note that if you win, there may be more than one entry in the primary index which matches, as we will see with our example, which goes like this:

- (1) Convert <https://www.w3.org/TR/xml/>
into urlkey: org,w3/tr/xml
- (2) In the primary index, we find on lines 843315 and 843316:

```
org,w3/tr/tr.xml cdx-00253.gz 557238519 185309
org,w3/wai/videos/standards-and-benefits/ja
cdx-00253.gz 557423828 182738
```
- (3) Extract 185309 bytes from cdx-00253.gz beginning at offset 557238519 and unzip the resulting block of 3000 lines
- (4) Binary search therein finds the following lines, beginning at line 1518:

```
org,w3/tr/xml
20210613173657
{"url": "https://www.w3.org/TR/XML/", "mime": "text/html", "mime-detected": "text/html", "status": "301", "digest": "LQRWZ7SMYYGCL55UJSVAS3BY64YNZ4DQ", "length": "743", "offset": "27241472", "filename": "crawl-data/CC-MAIN-2021-25/segments/1623487610196.46/crawldiagnostics/CC-MAIN-20210613161945-20210613191945-00275.warc.gz", "redirect": "https://www.w3.org/TR/xml/"}  

org,w3/tr/xml
20210613173657
{"url": "https://www.w3.org/TR/xml/", "mime": "text/html", "mime-detected": "application/xhtml+xml", "status": "200", "digest": "AOMNGHUQLUKLHHWBNU7MOVXKIUX522W", "length": "55091", "offset": "968583998", "filename": "crawl-data/CC-MAIN-2021-25/segments/1623487610196.46/warc/CC-MAIN-20210613161945-20210613191945-00371.warc.gz", "charset": "UTF-8", "languages": "eng"}
```

The first of these lines is the entry for <https://www.w3.org/TR/XML/>, which resulted in a redirection (status code 301) when retrieved, and the record of this is therefore contained in the non-200 archive component, indicated by the “crawldiagnostics” part of the filename.

The second line, with a status code of 200, is the one we were looking for, with “warc” in its filename.

The format of the primary index lines is simple:

urlkey<space>timestramp<space>JSON-array

In addition to the WARC filename, offset and length which allow direct access to the complete response, the JSON-array always contains the following features:

- “url”: as contained in the successful HTTP request
- “status”: HTTP response status code
- “mime”: from the Content-Type HTTP response header
- “digest”: a SHA1 hash of the response payload.

HTML responses have two additional computed features:

- “charset”: from the Content-Type HTTP response header
- “mime-detected”: ‘sniffed’ from the response body itself, using Apache Tika [2]
- “languages”: computed from the HTML using CLD2 [27]. May be absent, or contain up to three, comma-separated, lower-case ISO three-letter language codes.

3 BACKGROUND

There are relatively few reports of large-scale longitudinal studies of the Web based on data from Common Crawl. As was the case with our own earlier work [28], which was based on only two Common Crawl archives, such studies often draw on only a small number of archives. This is likely to be at least in part because of the large amount of processing required to tabulate some phenomenon from a number of multi-TB archives.

For example although Chapuis et al. [5] used seven archives from between 2016 and 2019, they commented that “Parsing the content of all the webpages contained in a snapshot [archive] of CC is time consuming”, so they proceeded to extract only 1% of those seven archives for use in most of their analysis. They don’t discuss the question of the mechanism by which that 1% was chose or how representative it was, but it does appear from their code archive that it was a single segment. Similarly, Luciano and Viviani [22] say “[B]oth downloading and analyzing the Common Crawl are time-consuming and costly endeavors” and go on to sample only 1% of a single archive.

In another recent example, a study of HTML validity [17], Florian and Stock appear to have used Common Crawl indices to identify candidate pages for analysis, but only actually downloaded around 1.8 million pages per archive from one archive per year from 2015 through 2022, that is, less than .1% of each.

Other work, such as [15], [25] and [7], uses only (parts of) single Common Crawl archives or even, as in [14], despite actually being focused on testing representativeness, don’t specify which or how many Common Crawl archives are used, only the (small) number of actual pages processed or analysed.

I’m have not been able to find any previous work which explicitly addresses the particular issues that concern us here, namely measuring the representativeness of individual segments and using Last-Modified headers to see further into the past.

On the general issue of Common Crawl’s goals, policies and use in the recent explosion of Large Language Models, a recent

Table 3: Sample from whole-archive mime tabulation from 2019-35

frequency	mime	mime-detected
2,232,464,436	text/html	ditto
650,577,285	text/html	application/xhtml+xml
40,022,222	unk	text/html
3,985,789	application/atom+xml	ditto
3,879,977	application/pdf	ditto
3,741,189	image/jpeg	ditto
2,741,054	unk	application/xhtml+xml
2,488,581	application/rss+xml	ditto
1,565,481	text/xml	application/rss+xml
1,229,831	text/plain	ditto

research paper from Mozilla [4] is the most comprehensive analysis I've seen.

4 PART 1: MEASURING AND EXPLOITING SEGMENT REPRESENTATIVENESS

4.1 Methodology

4.1.1 Analysing archive properties. Our first hypothesis is that by measuring the correlation between the distribution of features of the individual segments in an archive and their distribution in the archive as a whole, we can obtain a measure of the extent to which each segment in a Common Crawl archive contains a representative sample of the whole of that archive.

To test this we would like an easily available but rich property of the data whose overall distribution can be compared to its distribution in each segment.

Features found in the index satisfy the easily available requirement. We start by testing the use of a composite property based on the “mime” and “mime-detected” features.

For each of the four archives, we first tabulated the frequency of such pairs from the index entries for all successful retrievals, simplifying slightly by replacing a “mime-detected” value identical to the “mime” value with **ditto**. We did this for each segment first, and then merged the results to get a tabulation for the complete archives.

For example, Table 3 shows the top 10 pairs from the 2019 archive.

For each archive, we then created a merged tabulation for the top 100 pairs in that archive, with the total count and the count from each segment. Table 4 shows a small extract from this 101 x 101 table for 2019:

Two aspects of this example deserve comment:

(1) **The ‘nan’ in the middle** This means there was some data missing. Since we used the top 100 pairs from the whole corpus when taking counts from the individual segments, there are a few cases where a given pair from the whole-archive-top-100 did not occur at all in a given segment. Such drop-outs are recorded as ‘nan’ (not a number). The cutoff at 100 was chosen to keep this from happening very often, so in fact the four years had only 6, 5, 4 and 1 cases of this

Table 4: Example counts from merged mime tabulation for 2019-35

mime pair	whole archive rank	archive count	segment counts		
			71	72	73
text/plain application/mbox	52	37711	435	364	397
application/octet-stream	53	37414	354	nan	2
application/x-tika-msoffice					
application/octet-stream	54	36352	651	248	345
text/x-log					

Table 5: Example rank correlations

	whole archive	segments		
		71	72	73
whole archive	1.	0.947	0.949	0.937
segment 71	0.947	1.	0.896	0.894
segment 72	0.949	0.896	1.	0.899
segment 73	0.937	0.894	0.899	1.

respectively out of 10,000, and the impact of the missing entries is negligible.

(2) **The red, bold-face pairs of counts** These highlight mismatches between the ranks of 53 and 54 in the whole archive for the 2nd and 3rd media pairs and their counts in segments 71 and 73.

Those two mis-matches suggest a way to measure the representativeness of the individual segments: the rank correlation between the columns of this table. We use rank correlation because, as is already evident from Table 3, the distributions involved are far from normal.

We used the `stats.spearmanr` function in the Python Scipy library [29] to compute a full 101 x 101 array of the rank correlation between every possible pair in the 100 x 101 tabulation of counts, using the ‘omit’ policy for dealing with missing (‘nan’) cells.

Table 5 shows a small extract, using the same columns as in Table 4, from the resulting correlation⁴ array for 2019. For what it’s worth, the p-values produced by the `stats.spearmanr` function were tiny, on the order of 10^{-60} or less. In subsubsection 4.2.1 below we’ll provide confidence intervals, which give a better sense of the reliability of the correlation values.

The results are necessarily symmetrical around the self-correlation diagonal. The pattern shown in this example, where the correlation between each segment and the whole is noticeably higher than the cross-correlations with any other segment, is repeated throughout the whole tabulation.

4.1.2 Using one property as a proxy for another. Our goal is to explore the possibility of using a property we have complete data for to predict the best segment(s) to use in place of the whole. This may be particularly useful for studying properties for which we *don’t* have complete data. We will illustrate this using properties based on two other features from the index for 2019-35:

⁴We’ll just say “correlation” from now on, but all numbers described as such are actually *rank* correlations

Table 6: Descriptive statistics for segment-vs-whole rank correlations between mime property distributions

Archive	N	min	max	mean	variance
2019-35	100	0.898	0.962	0.932	0.0002
2020-34	100	0.891	0.958	0.929	0.0002
2021-31	100	0.873	0.965	0.928	0.0004
2023-40	100	0.853	0.956	0.926	0.0004

- (1) **language property:** For `text/html` responses, Common Crawl uses the CLD2 tool to populate the “languages” field in the index with up to 3 language codes, in rank order, from which we used only the first. As for the mime case, we produced a 101×101 table of rank correlations with respect to the number of occurrences of each language in the archive as a whole and in each segment. We included only the top 100 languages in order to keep the number of NaNs to a reasonable level.
- (2) **length property:** The index includes the (zipped) “length” of the response, which we tabulated as a percentile. Once again this gave us a 101×101 table of rank correlations.

For the 2019-35 data, we then tested how successful using one or more of the best (in terms of correlation with the whole) segment for one property was, in terms of those same segments’ performance on a different property (again, in terms of correlation with the whole). We report this as a percentile *vis-a-vis* the range of values for all 100 segments (as shown for the mime property in Table 5).

For example, taking the top five segments in terms of correlation with the whole for the merged mime property (the **basis**) and averaging those segments’ correlation with the whole for the language property (the **target**) gives 57.7. This is better than the overall mean of 54.6 for individual segments vs. the whole on that measure, by about .4 of a standard deviation.

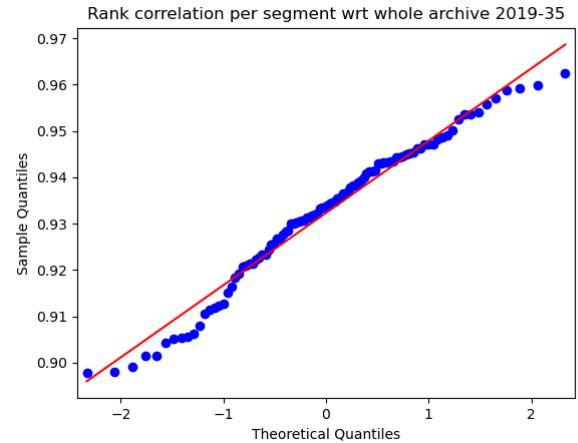
We tested each of our three measures as the basis against the other two as target, giving a total of six pairings. For each of these pairs we tested each of the top ten basis segments again the target. We then tabulated the percentile performance of the top one, and averages of the top two, three, ...ten.

Note that this is only a simulated test of finding proxies for properties that are *not* in the index. We did this precisely because this way we were able to compare our results across a wider range of data than working with *actual* whole-archive results, which would have taken more computational resources than we had available.

4.2 Results

4.2.1 Mime property. As noted above, we looked at the full 101×101 correlation matrix for the distribution of the mime property, derived from the index, tabulated for each whole archive and its 100 individual segments.

Although the cross-correlations *between* segments have some interesting properties, for our purposes we will focus here on the 100 correlations between each segment and the whole archive. Table 6 tabulates values from Scipy’s `stats.describe` as applied to these for our four archives.

**Figure 1: QQ plot for segment-vs-whole correlations, 2019-35.**
Based on the distribution of the mime property

This all looks good from the point of view of our ultimate goal, that is, to identify segments in any archive that accurately reflect the archive as a whole, in that the ‘best’ segments each year have very high correlations (>0.95) with the whole archive.

The distribution of the correlations between segments and the whole archive is reasonably close to normal in all years, with Shapiro-Wilks values of 0.972, 0.975, 0.945, 0.84, $p \leq 0.05$ in all cases.

The QQ-plot produced by the StatsModels `api.qqplot` [26] for the 2019-35 data is shown in Figure 1 and the other three are similar, although the 2023 plots show 5 outliers at the bottom end, consistent with the somewhat lower Shapiro-Wilks result.

A histogram with mean and standard deviations marked for 2019-35 is shown in Figure 2.

The same data is plotted per-segment, with the overall mean, in Figure 3.

And finally, same data again, ordered by correlation and with 95% confidence intervals shown, in Figure 4. This certainly suggests that the best 5–10 segments are likely to be better choices as proxies for the whole archive, and even more likely that the *worst* segments should be avoided.

Appendix B gives a complete tabulation of segment rank for all four years, using the three properties described above (mime, language and length).

4.2.2 Testing predictions across properties: Mime, language and length. We proceeded to see if the correlations computed as described in the previous section can identify one or more segments from a Common Crawl archive which are good proxies for the whole.

In particular, we can now determine what property, available from the index, is best for predicting the segments to use for *other* properties, as a prelude to working with properties which are not available from the index. Figure 5 shows, in the form of heatmaps, all possible pairings of three properties being used as the basis for predicting one another as targets, computed as described above

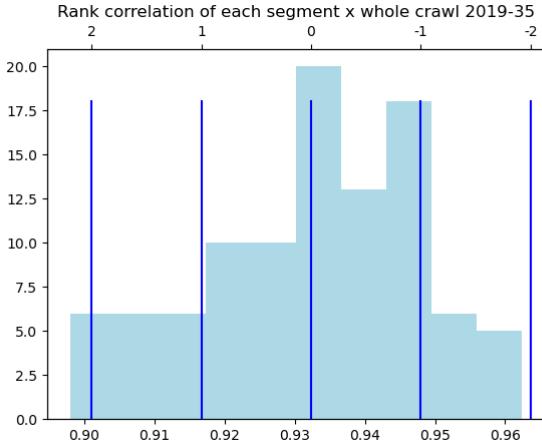


Figure 2: Histogram for segment-vs-whole correlations, 2019-35. Based on the distribution of the mime property. Blue vertical lines show standard deviations from the mean.

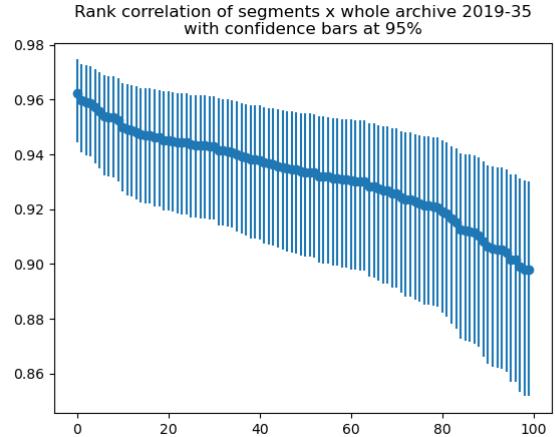


Figure 4: Segment-vs-whole correlations, ordered by correlation, with 95% confidence intervals, 2019-35. Based on the distribution of the mime property. Calculated using the atanh approach as described in [11]. The worst error bar is just disjoint from the best.

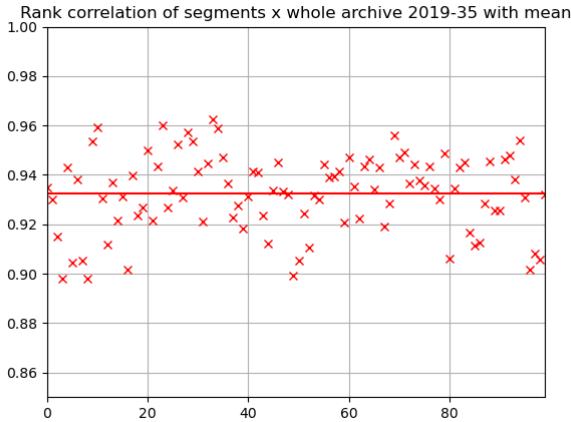


Figure 3: Segment-vs-whole correlations, ordered by segment, 2019-35. Based on the distribution of the mime property

in subsubsection 4.1.2. For example the cell in 5th column of the second row of the heatmap, with value 57.7, tests how well we can predict a good subset of the segments from the 2019-35 archive to use to study the distribution of the language property in that entire archive, using the identity of the top 5 segments from that archive in terms of the correlation between their distribution of the *mime* metric with its distribution in the entire archive.

We can see that the predictions are not perfect, but we can learn what we need from the table as a whole: The language property is both better on average and more consistent. The length property is the least reliable, although exactly *why* it's so much worse at

predicting the mime property than the language property deserves further investigation.

It's less clear what the best choice for N is. Overall it seems that almost any choice in the 1–5 range will be OK. We'll look at that in more detail in the next part, when we trial this approach in a case where we really don't have all the target data available so we really do need to use a proxy.

5 PART 2: USING LAST-MODIFIED DATE TO EXPLORE URI LENGTH

5.1 Methodology

Common Crawl only started releasing archives in 2008, but we can still get at least some insight even further back into the history of the Web from even the most recent archives, by exploiting the information in the Last-Modified HTTP response header. Although this is optional, it is present in around 17% of the successful responses for 2019-35. There are around 600 million responses with a Last-Modified header out of around 3.4×10^9 in total. The earliest credible values archive are from the late 20th century.

According to the HTTP specification [16], the Last-Modified header value must be in one of several standard textual formats, preferably what is defined there as **HTTP-date**. However not all Web servers conform to that requirement. We allowed a certain limited amount of flexibility, for example in the (mis)placement of "GMT", but still ended up rejecting about .01% as unusable as written, and a further .1% because they were not credible (too early or in the future).

The accepted values were then converted into POSIX time format [18], for easier sorting and comparison.

Predict	N										Basis			
	1	2	3	4	5	6	7	8	9	10	Avg	StDev	Avg	StDev
len by mime	70.2	68.2	62.4	46.8	50.3	49.7	53.6	51.6	54.8	56.5	56.4	7.6	54.6	6.6
nl1 by mime	49.2	44.2	45.3	54.2	57.7	52.1	57.3	55.2	58.9	53	52.7	4.8		
mime by len	58.1	39.5	44	41	43.1	39.2	41.4	45.1	41.3	42.2	43.5	5.2	51.3	8.8
nl1 by len	62.9	57.8	63	57.5	60.4	58	58.8	60.1	55.4	57.5	59.1	2.3		
mime by nl1	69.8	70	55.6	63.4	68	66.1	58.9	57.4	61.5	62.6	63.3	4.8	66.1	5.1
len by nl1	70.8	70.2	67	69.6	71.7	71.2	73.1	70.3	62.5	61.8	68.8	3.7		
Average	63.50	58.32	56.22	55.42	58.53	56.05	57.18	56.62	55.73	55.60	57.32	2.30		

Figure 5: Heat map for prediction percentiles, 2019-35. The first column lists what property is being predicted by what: length (“len”), mime and language (“nl1”). For example the first row gives the results for predicting what segment(s) should be used to model the distribution of the length property (the **target**) based on the best N segments from the distribution of the mime property (the **basis**). The figure in a given cell is the percentile rank of the average of the segment-to-whole correlations from those segments in the target identified as the best N segments in the basis. The heat map colouring is per *row*, that is, across all the predictions named in the first column as we vary the number used (N). The black-margin cells highlight the best combination of basis and N for each target. The final two columns give mean and standard deviation over all the predictions for a given basis, i.e. the based on the twenty cells to the left.

To complete our test of the proxying hypothesis, we wanted to use the 2023-40 archive, for which we did *not* have the complete archive, only the index, to start with.

Since we did have the full archive for 2019-35, we tested one further property as target, namely the distribution of frequencies of Last-Modified headers by year, to choose a proxy for our study of 2023-40. Figure 6 shows the results of a further round of testing in which the Last-Modified frequency was the target with the same three properties as before as basis.

This shows somewhat less good results overall, but does confirm that “language” is the best predictor. We chose to use $N = 2$ to reduce the risk of hitting a bad segment by accident, and fetched just the top two segments as measured by the “language” property from the 2023-40 index, that is, segments 56 and 12 (see Appendix B).

We extracted and processed the Last-Modified values from those two 2023-40 segments using the same method described above for 2019-35. Taken together they contain around 12 million responses with a Last-Modified header out of around 69 million in total, about 17%, the same ratio as observed for the 2019-35 data. We then sorted and tabulated the Last-Modified dates by year, by month within 2023 and by day within September 2023.

Figure 7 shows the number of headers found by year. The years before 2000 are clearly very poorly represented, and are not included in our subsequent analysis of URI length. 2005 also is badly distorted from what looks like an anomaly in the Common Crawl data gathering pipeline or the seeds from which it starts. See Appendix A for details of the evidence that this is an anomaly and the corrective action taken, which affects all data and figures after Table 6 and Figure 7 and the discussion thereof.

A semi-log plot is used in Figure 7 because of the rapid fall-off in pages with Last-Modified headers with values before 2023: over

10 million in 2023, only 1.3 million in total from earlier years. More details on this are provided in subsection 5.2.

In order to look at URI length over time, we tabulated the overall length of each URI from 2023-40 for which we have a Last-Modified date, as well as the lengths of its parts (scheme, netloc, path, and query) and three additional measures:

- (1) **idna**: whether or not a non-ascii netloc was present, encoded using punycode
- (2) **path%**: for URIs with a non-empty path, the number of percent-encoded characters therein
- (3) **query%**: for URIs with a non-empty query, the number of percent-encoded characters therein

5.2 Results

Figure 8 compares the proxied results from 2023-40 with both the full-archive results from 2019-35 and a comparable best-two segment result from 2019-35.

This does suggest that the chosen proxy *is* a representative of the data as a whole. Particularly reassuring is the conformance of the 2023-40 proxy curve to the 2019-35 whole archive curve, as the individual pages in those two years are almost all different: the overlap between the two is less than .4% as measured by the Common Crawl overlap statistics.

5.2.1 URI lengths. The actual change in the length of URIs, and their component parts, is shown in Figure 9. Figure 10 shows the change in the path and query components separately. It appears that our hypothesis was wrong, and that URI length is increasing only slowly, and the change is more due to growth in path length than query length.

5.2.2 Last-minute Last-Modified values. Something interesting emerges if we explore the Last-Modified values at a detailed granularity. We should have anticipated this given what we know about the change

Predict	N										Basis			
	1	2	3	4	5	6	7	8	9	10	Avg	StDev	Avg	StDev
lmh by mime	73.7	36.9	49.5	52.6	57.2	56.3	60.8	63.2	63.3	63.5	57.7	9.4	57.7	9.4
lmh by len	70.8	77.4	64.3	63.8	66.7	65.5	65.8	65.9	63.9	63.7	66.8	4.1	66.8	4.1
lmh by nl1	88	75.4	73.3	76.9	74.3	74.5	77	78.5	76.6	77.1	77.2	3.9	77.2	3.9
Average	58.4	47.9	47.5	49.3	50.8	50.6	52.7	53.9	53.2	53.6	51.8	3.1		

Figure 6: Heat map for prediction percentiles, 2023-40. Last-Modified header count per year (“lmh”) predicted by mime, language and length. Layout the same as Figure 5. As there is only one target, the only the black border now is the best cell overall.

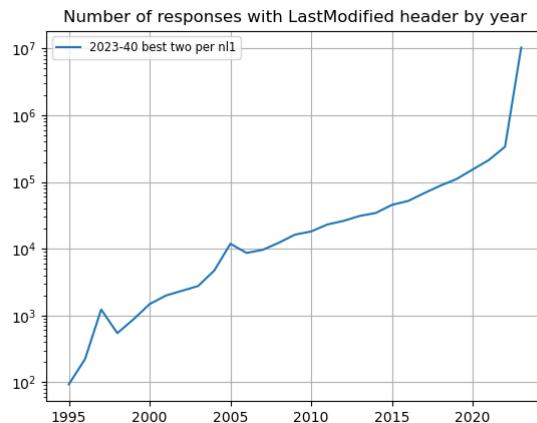


Figure 7: Last-Modified header counts by year. Based on uncorrected data from the 2023-40 archive. Semi-log plot for the y-axis.

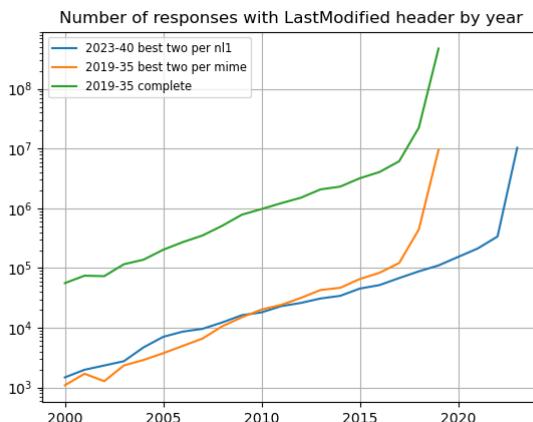


Figure 8: Comparing Last-Modified header counts by year. Based on data from the corrected 2019-35 and 2023-40 archives. Semi-log plot for the y-axis.

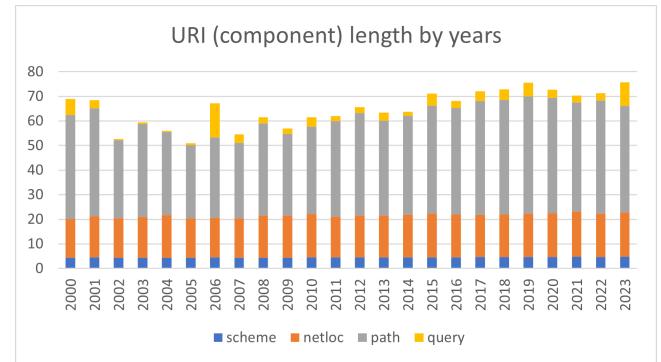


Figure 9: URI length, divided into component lengths, by year. Based on data from the 2023-40 archive.

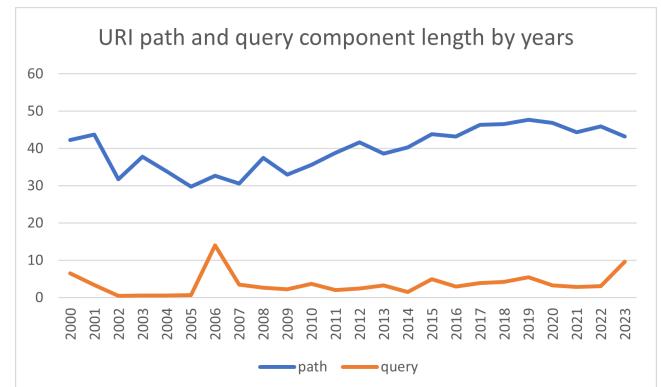


Figure 10: URI path and query length, by year. Based on data from the 2023-40 archive.

from a human-authored web to a just-in-time machine-generated web, and indeed some of our previous work [13] suggested Common Crawl gives evidence of it.

Figures 11 and 12 focus in on the year and month our proxy segments for 2023-40 were sampled.

The vast majority of the pages have Last-Modified dates on the two days each of those proxy segments were crawled, 21 and 29 September 2023.

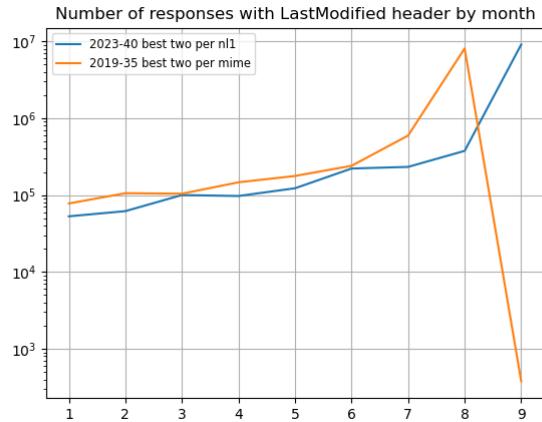


Figure 11: Last-Modified header counts by month. Based on data from the 2023-40 archive. Semi-log plot for the y-axis.

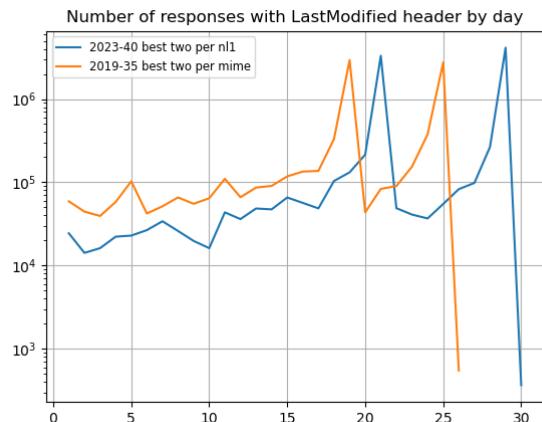


Figure 12: Last-Modified header counts by month. Based on data from the 2023-40 archive. Semi-log plot for the y-axis.

In order to check this further, we can compare the Last-Modified data for pages crawled on those key days with the crawl timestamp, to see how many of them were being created on the fly in response to the crawler request. Figure 13 does exactly that, by converting the crawl timestamps to 10-digit POSIX timestamps and subtracting them from the Last-Modified times for all the pages for which we have Last-Modified data that were crawled on 21 or 29 September.

53% of the offsets are 0.0, 70% are within three seconds of the crawl time. The marginal numbers are interesting in their own right—they are at exactly -5, -4, -1 hours earlier and +1 and +2 hours later than the crawl time. That's what we would expect from just-in-time pages from various likely areas if their Last-Modified values were expressed as per their local timezone without being

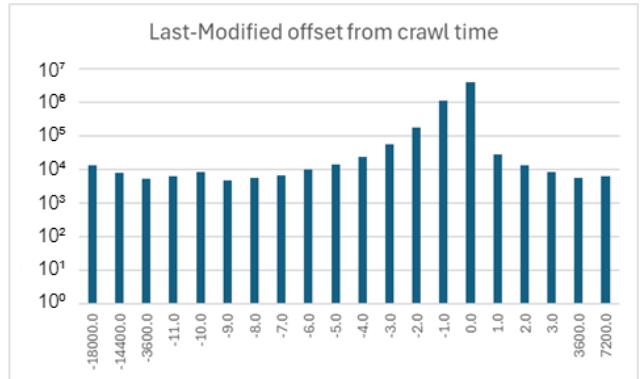


Figure 13: 20 most frequent Last-Modified time offsets from crawl time, in seconds. Based on data from the 2023-40 archive. Overall N is 7,405,211, of which these account for 5,442,578 = 74%. Negative column labels correspond to Last-Modified times *before* their crawl time, positive ones to L-M times *after*. Semi-log plot for the y-axis.

converted to UTC and did not contain a timezone indication. For example, -14400, equivalent to 0 in UTC-4, is what we'd expect from the East Coast of North America if local times in September were reported without an explicit "EDT" (Daylight Savings ended on 5 November in most of the USA).

6 DISCUSSION

6.1 Ranking segments as proxies for the whole

For each of the four archives we've tested, we've shown that overall the segmentation does a pretty good job of producing subsets which closely resemble the whole archive with respect to the distribution of media types: the *worst* rank correlation for a segment is .873, and most are over 0.9. We've done this in a way that can be easily applied to other Common Crawl archives. We plan to produce and publish a list of segment rankings with respect to several properties for every archive since January 2019, based on this approach.

We hope the availability of these rankings will in turn make it much easier to carry out longitudinal studies using only 1% or 10% of a series of full archives, bringing such work into reach for those without access to large high-performance clusters.

6.2 Explaining the growth in URI length

The upward blip in the query in 2006 deserves further investigation, but with N still less than 10,000 it's probably just due to the lumpiness of random processes: after removing all the URIs from two domains with queries which were sampled more than 100 times and whose average query length is greater than 100, the mean query length is down to a similar value to that of the surrounding years.

Once we paid more attention to the predominance of last-minute Last-Modified values, it looks like the increase in overall size and of the query component for 2023 is in fact from pages with very small offsets between Last-Modified date and crawl time, compared

more ‘normal’ offsets. More investigation of more data is needed to get a clear picture of what’s going on.

6.3 Last-minute Last-Modified values

It is interesting to note that up until a few days before the crawl date, the upward slope of the day-by-day curves in Figure 12 is similar to that of the month-by-month curves Figure 11, but slower than that of the year-by-year curves Figure 8. It would be interesting to compare those slopes to whatever the literature has to say about the actuarial profiles of human-authored pages versus created-on-demand pages. Or correlation with the amount of JavaScript...

ACKNOWLEDGMENTS

Access to the Cirrus UK National Tier-2 HPC Service at the Edinburgh Parallel Computing Centre (<http://www.cirrus.ac.uk>) used in this work was supported by an EPSRC and UKRI HPC Access awards to Henry S. Thompson.

The work reported here has been stimulated by a number of MSc projects over the last few years. Particular thanks in the regard are due to Jian Tong, Lukasz Domanski and Jingrui Chen.

Thanks to Sebastian Nagel of Common Crawl for many prompt and helpful replies to many emails over the year, and to Greg Lindahl of Common Crawl and Tom Morris for more recent help with consistency problems in the index and the challenges of increasing load on the Common Crawl servers.

All use of Common Crawl archives and indices reported on in this work conformed to the Common Crawl Terms of Use [10].

REFERENCES

- [1] Amazon. [n. d.]. Amazon Web Services (AWS) Open Data Sponsorship Program. Web document. <https://aws.amazon.com/opendata/open-data-sponsorship-program/> Retrieved: 22 November 2023.
- [2] Apache Software Foundation. [n. d.]. Content analysis toolkit. Web document. <https://tika.apache.org/> Retrieved 7 December 2023.
- [3] Internet Archive. [n. d.]. Sort-friendly URI Reordering Transform. Web document. http://crawler.archive.org/articles/user_manual/glossary.html#surt Retrieved: 29 November 2022.
- [4] Stefan Baack and Mozilla Insights. 2024. Training Data for the Price of a Sandwich. Web document. <https://foundation.mozilla.org/en/research/library/generative-ai-training-data/common-crawl/> Retrieved: 25 February 2024.
- [5] Bertil Chapuis et al. 2020. An Empirical Study of the Use of Integrity Verification Mechanisms for Web Subresources. In *Proceedings of The Web Conference 2020* (Taipei, Taiwan) (*WWW '20*). Association for Computing Machinery, New York, NY, USA, 34–45. <https://doi.org/10.1145/3366423.3380092>
- [6] Jingrui Chen. 2021. *A Survey on HTTP cookies: Do large Internet companies collect more information from users?* MSc dissertation. University of Edinburgh.
- [7] Aatish Chiniyah, Ayaz Chummun, and Za Burkutlyiyid. 2019. Categorising AWS Common Crawl Dataset using MapReduce. In *2019 Conference on Next Generation Computing Applications (NextComp)*. 1–6. <https://doi.org/10.1109/NEXTCOMP.2019.8883665>
- [8] Common Crawl. 2022. Common Crawl - Get Started. Web document. <https://commoncrawl.org/get-started/> Retrieved: 22 November 2023.
- [9] Common Crawl. 2023. Common Crawl - Open Repository of Web Crawl Data. Web document. <https://commoncrawl.org> Retrieved: 22 November 2023.
- [10] Common Crawl. 2024. Common Crawl - Terms of Use. Web document. <https://commoncrawl.org/terms-of-use> Retrieved: 24 February 2024.
- [11] Nick Cox. 2011,2023. How to calculate a confidence interval for Spearman’s rank correlation. Web document. <https://stats.stackexchange.com/a/18904> Retrieved: 6 December 2023.
- [12] Peter Deutsch. 1996. *GZIP file format specification version 4.3*. Internet RFC. IETF. <https://www.ietf.org/rfc/rfc1952.html> Retrieved: 22 November 2023.
- [13] Lukasz Domanski. 2020. *Analysing Common Crawl - Efficient and Cost-Effective Processing of Large-Scale Data*. MSc dissertation. University of Edinburgh.
- [14] Yuheng Du et al. 2017. Representativeness of latent dirichlet allocation topics estimated from data samples with application to common crawl. In *2017 IEEE International Conference on Big Data (Big Data)*. 1418–1427. <https://doi.org/10.1109/BigData.2017.8258075>
- [15] Julian Eberius et al. 2015. Building the Dresden Web Table Corpus: A Classification Approach. In *2015 IEEE/ACM 2nd International Symposium on Big Data Computing (BDC)*. 41–50. <https://doi.org/10.1109/BDC.2015.30>
- [16] Roy T. Fielding and Julian Reschke. 2014. Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests. RFC 7232. <https://doi.org/10.17487/RFC7232> Retrieved: 22 November 2023.
- [17] Florian Hantke and Ben Stock. 2022. HTML violations and where to find them: a longitudinal analysis of specification violations in HTML. In *Proceedings of the 22nd ACM Internet Measurement Conference (Nice, France) (IMC '22)*. Association for Computing Machinery, New York, NY, USA, 358–373. <https://doi.org/10.1145/3517745.3561437>
- [18] IEEE. 2017. The Open Group Base Specifications Issue 7, 2018 edition IEEE Std 1003.1-2017: POSIX.1a. Web document. https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap04.html#tag_04_16 Retrieved: 25 November 2023.
- [19] International Internet Preservation Consortium. 2017. Web document. <https://iiipc.github.io/warc-specifications/specifications/warc-format/warc-1.1/> Retrieved: 29 November 2022.
- [20] Ilya Kreymer. 2015. Web document. <https://github.com/ikreymer/pywb/wiki/CDX-Index-Format#zipnum-sharded-cdx> Retrieved: 22 November 2023.
- [21] Ilya Kreymer. 2015. Announcing the Common Crawl Index. Web document. <https://commoncrawl.org/blog/announcing-the-common-crawl-index> Retrieved: 22 November 2023.
- [22] Alexandra Luccioni and Joseph Viviano. 2021. What’s in the Box? An Analysis of Undesirable Content in the Common Crawl Corpus. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (Eds.). Association for Computational Linguistics, Online, 182–189. <https://doi.org/10.18653/v1/2021.acl-short.24>
- [23] Eve Maler, Jean Paoli, et al. 2008. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. Fifth Edition of W3C Recommendation. W3C. <http://www.w3.org/TR/xml/> Retrieved: 29 November 2022.
- [24] Sebastian Nagel. 2022. Common Crawl (Getting Started). Web document. <https://commoncrawl.org/the-data/get-started/> Retrieved: 29 November 2022.
- [25] Alexander Panchenko et al. 2018. Building a Web-Scale Dependency-Parsed Corpus from CommonCrawl. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Nicoletta Calzolari et al. (Eds.). European Language Resources Association (ELRA), Miyazaki, Japan.
- [26] Statsmodels Project. [n. d.]. statistical models, hypothesis tests, and data exploration. Web document. <https://www.statsmodels.org/stable/index.html> Retrieved: 28 November 2022.
- [27] Dick Sites. 2013. Compact Language Detector 2. Web document. <https://github.com/CLD2Owners/cld2> Retrieved 7 December 2023.
- [28] Henry S. Thompson and Jian Tong. 2018. Can Common Crawl Reliably Track Persistent Identifier (PID) Use Over Time. In *Companion Proceedings of the The Web Conference 2018* (Lyon, France) (*WWW '18*). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1749–1755. <https://doi.org/10.1145/3184558.3191636>
- [29] Pauli Virtanen et al. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17 (2020), 261–272. <https://doi.org/10.1038/s41592-019-0686-2>

A DETECTION AND CORRECTION OF LAST-MODIFIED ISSUE

As noted above in the discussion of Figure 7, the frequency of Last-Modified dates in 2005 stands out from the surrounding years. Table 7 shows the details of this in increasingly fine granularity, to the surprising point where we see that the whole difference is down to a single day.

Further inspection of the Last-Modified values for that day revealed that the huge increase on that day was due to a single Last-Modified value, namely Sun, 24 Apr 2005 04:29:37 GMT.

We confirmed the anomalous nature of anything like this by tabulating the frequency of all the Last-Modified values found in the 2019-35 archive in 10000-second intervals, by counting the first 6 digits of the corresponding 10-digit POSIX time value. Figure 14 shows the 10 most frequent counts for any such interval in 2005

Table 7: Evidence for anomaly in Last-Modified frequency
By year, month and day. The column in bold is the anomaly. Based on data from the uncorrected 2019-35 archive.

Source	counts				
	2003	2004	2005	2006	2007
Years	115926	138295	567693	272565	351429
Months in 2005	Feb	Mar	Apr	May	Jun
Days in 2005-04	17114	15083	378061	14640	19119
	22	23	24	25	26
	362	215	365113	1167	554

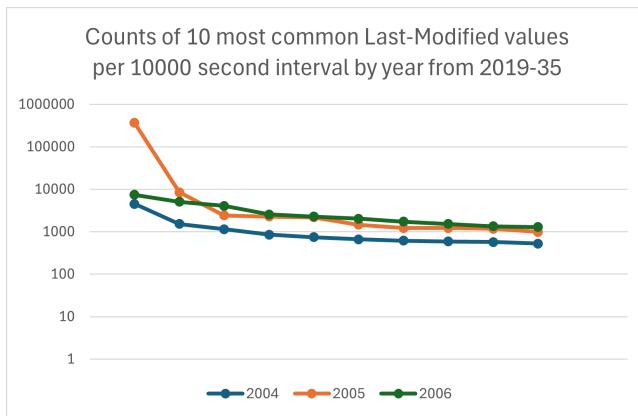


Figure 14: Last-Modified frequency by 10000 second ranges
The left-most orange point is for the range including 1114316977.
Based on data from the uncorrected 2019-35 archive. Semi-log plot for the y-axis.

Table 8: Zooming in on anomaly in Last-Modified counts Top row for each archive shows counts for the most common interval for Last-Modified values, bottom row for the second-most common. The figures in bold are for the interval containing 1114316977. Based on data from the uncorrected 2019-35 archive.

Archive	Last-Modified year		
	2004	2005	2006
2019-35 (whole)	4511	364934	7400
	1521	8521	5047
2023-40 (6 segs only)	215	13408	857
	187	824	556

and similar values for the surrounding years, that is, not necessarily for the same interval, but the *same-ranked* interval. The outlier, by nearly two decimal orders of magnitude, is the interval containing 1114316977, the 10-digit POSIX version of the anomalous value.

Table 8 zooms in on counts for full 10-digit Last-Modified values from the year in question, 2005, and one year on either side.

This unique value accounts for all but 9 cases of its corresponding 6-figure value for 2019-35 and all but 3 for 2023-40. This 10-figure value never appears in the other two years in either archive. This

level of frequency for a single exact Last-Modified value is unprecedented. The next most common full 10-digit LM in the 6-segment 2019-35 archive data for these three years occurs only 7329 times, in the 6 segments of 2023-40 only 805 times, so we're looking at factors of 49 and 15.

On this basis I judged that the twin coincidences of a super-unlikely over-count for a single LM date, across over 900 domains in 2023-40, over 31000 in 2019-35 (mostly .ru, .ua, .su, .am, .kz, .xmp1ai == pφ), and the fact that it occurs in *different* archives from mostly *different* domains (more than half the domains from 2023-40 don't show up again in 2019-35) is an indication of a problem somewhere.

The subsequent analyses reported here are therefore based on Last-Modified data accordingly are based on data with all entries with the 1114316977 Last-Modified value removed, amounting to 378,330 out of approximately 521×10^6 .

I'm working with the Common Crawl technical staff to try to find the cause, but as of this writing without success.

B SEGMENT RANKING TABLES

The accompanying table gives top-10 complete segment rankings, sorted on correlation of the mime property discussed above in subsubsection 4.2.1 and divided into deciles, for the first three years reported here. Full rankings, as well as the index for 2019-35 with Last-Modified times added, will be published as soon as I can find a free place to host them.

Table 9: Segment ranks (best-to-worst) based on media type distribution correlations

rank	Archive		
	2019-35	2020-34	2021-31
1	33	79	49
2	23	71	83
3	10	34	87
4	34	83	81
5	28	73	62
6	69	38	93
7	94	20	77
8	29	88	86
9	9	44	94
10	26	45	99