MementoMap Framework for Flexible and Adaptive Web Archive Profiling

Sawood Alam
Old Dominion University
Department of Computer Science
Norfolk, Virginia, USA
salam@cs.odu.edu

Michele C. Weigle
Old Dominion University
Department of Computer Science
Norfolk, Virginia, USA
mweigle@cs.odu.edu

Michael L. Nelson
Old Dominion University
Department of Computer Science
Norfolk, Virginia, USA
mln@cs.odu.edu

Fernando Melo
FCT: Arquivo.pt
Lisbon, Portugal
fernando.melo@fccn.pt

Daniel Bicho
FCT: Arquivo.pt
Lisbon, Portugal
daniel.bicho@fccn.pt

Daniel Gomes
FCT: Arquivo.pt
Lisbon, Portugal
daniel.gomes@fccn.pt

ABSTRACT

In this work we proposed *MementoMap*, a flexible and adaptive framework to summarize holdings of a web archive efficiently. We described a simple, yet extensible, file format suitable for *MementoMap*. We used the complete index of the Arquivo.pt comprising 5B mementos (archived web pages/files) to understand the nature and shape of its holdings. We generated *MementoMaps* with varying amount of detail from its *HTML* pages that have an *HTTP* status code of `200 OK`. Additionally, we designed a single-pass, memory-efficient, and parallelization-friendly algorithm to compact a large *MementoMap* into a small one and an in-file binary search method for efficient lookup. We analyzed more than three years of *MemGator* (a Memento aggregator) logs to understand the response behavior of 14 public web archives. We evaluated *MementoMaps* by measuring their *Accuracy* using 3.3M unique *URIs* from *MemGator* logs. We found that a *MementoMap* of less than 1.5% *Relative Cost* (as compared to the comprehensive listing of all the unique original *URIs*) can correctly identify the presence or absence of 60% of the lookup *URIs* in the corresponding archive while maintaining 100% *Recall* (i.e., zero false negatives).

KEYWORDS

Memento, Web Archiving, Archive Profiling, MementoMap

1 INTRODUCTION

Old Dominion University (ODU) runs MemGator [7] as a service to power many of our tools and services such as Mink [30], CarbonDate [14], WAIL [16], ICanHazMemento [33], and Memento-Damage [37]. We released MemGator [2] as an open-source tool for users to run locally to avoid too much traffic on a central aggregator service. Our service receives three aggregation lookup requests per minute on average. Due to this low traffic we do not yet use any prediction-based Memento routing or caching. We recently analyzed over three years of our MemGator logs and found that it has served about 5.2M requests so far. These lookups were broadcasted to 14 different upstream archives as a total of 61.8M requests. Only 5.44% of these requests had a hit, while the remaining 93.56% were either a miss or an error as shown in Table 3. If only there was a way to know a summary of the holdings of these web archives, we could have avoided many wasted upstream requests and had an overall better response time for clients.

MementoMap is a framework for profiling web archives and expressing their holdings in an adaptive and flexible way to easily scale. It is inspired by the simplicity of widely used `robots.txt` and `sitemap.xml` formats, but for a purpose other than search engine optimization. An example *MementoMap* is illustrated in Figure 2 in the format we propose. *MementoMap* allows wildcard-based partial *URI Keys* to enable flexibility in how detailed or concise one wants it to be depending on use cases, full or partial knowledge about the archive's holdings, and available resources. This can either be generated by the archives themselves or by a third party based on their external observations. We propose the "`mementomap`" link relation for its dissemination and discovery.

We used the complete index of Arquivo.pt - the Portuguese web-archive (PWA), spanning over 27 years, and more than three years of MemGator logs for evaluation. We found that a summarized *MementoMap* of less than 1.5% *Relative Cost* (as compared to the comprehensive listing of all the unique original *URIs*) can correctly identify the presence or absence of 60% of the lookup *URIs* in PWA without any false negatives (i.e., 100% *Recall*). We have open-sourced our implementation [3] under the *MIT* license terms.

2 BACKGROUND

The Internet Archive (IA)[1] is the first, largest, and most resourceful web archive with over 700B mementos (timestamped archived copies of web pages and files) as of January 21, 2019[2]. However, it is also the softest target for censorship and denial of service attacks [28]. It continues to be blocked in China [25] and Russia [22] for an extended period of time and has been blocked temporarily in many other countries such as India and Jordan [1, 18]. As a result, many web archiving related tools are increasingly adding support for Memento aggregators to consolidate archived resources from more than one web archive of varying scale to avoid single point of failure.

The Memento framework [39] defines uniform APIs for *TimeMap* and *TimeGate* endpoints to enable cross-archive communication. A *TimeMap* is a list of all mementos of an original URI (or *URI-R*) and a

[1]https://archive.org/
[2]https://twitter.com/brewster_kahle/status/1087515601717800960

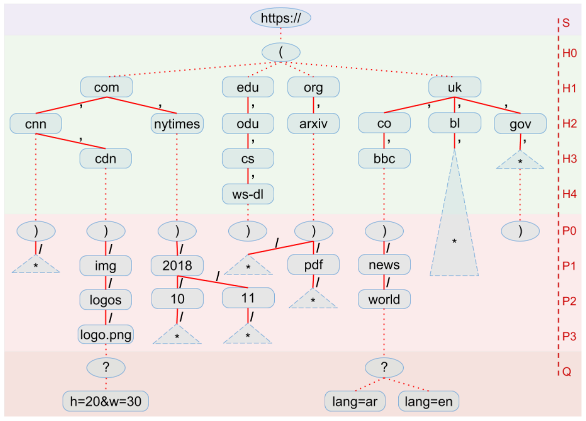


Figure 1: Visual Representation of SURT

TimeGate is a gateway to resolve to the closest memento of a *URI-R* w.r.t. a given *Datetime* and redirect to a Memento URI (or *URI-M*). With out-of-the-box Memento support in major archival tools and replay systems, many web archives have adopted the protocol. To avoid the need of every tool being configured and periodically updated to poll results from an ever-changing list of many known web archives, Memento aggregators were created to act like a single consolidated web archive to users and tools. Los Alamos National Laboratory (LANL)'s *Time Travel* [3] service is one such well-known aggregator that powers many tools and services. *MemGator* is our open-source Memento aggregator implementation that can be used locally as a CLI tool or run as a service for a drop-in replacement of the *Time Travel* service.

CDX (Capture inDeX) [26] is a *CSV*-like text file-based index format that has traditionally been used by the IA and was one of the primarily supported index formats of OpenWayback[4]. It is very rigid in nature and has a predefined list of fields that are not extendable. While working on this paper, we encountered a consequence of its limitations when we realized that the *MIME-Type* field was reused to record a different metadata to identify whether a record is a *revisit*. As a result the actual *MIME-Type* of the record would not be known without finding another entry in the index which the record is a *revisit* of. *CDXJ* [6] is an evolution of the classic *CDX* format. In this file format, lookup key fields (*URI-R* and *Datetime*) are placed at the beginning of each line which is followed by a single-line compact JSON [20] block that holds other fields that can vary in number and be extended as needed. Both of these formats are sort-friendly to enable binary search on file when performing lookups. The latter format is primarily used by archival replay systems including PyWB[5] and our InterPlanetary Wayback (IPWB) [29].

SURT (Sort-friendly URI Reordering Transform) [36] is used to canonicalize *URIs* and place together related *URIs* when sorted, which is important for efficient indexing. In a traditional URI the hostname parts are organized differently than paths. In the hostname section, the root of the Domain Name System (*DNS*) chain (i.e., the Top Level Domain, or *TLD*) comes at the end towards the right hand side while registered domain name portion and subdomain sections are placed towards the left hand side. In contrast, in the path section, the root path comes first followed by deeper nodes of the path tree towards the right side. As a consequence, if a list of three domain names `example.com`, `foo.example.com`, and `example.net` are sorted, the latter with a different *TLD* will sit in between the other two. As opposed to this the *SURT* of "`Www.Foo.Example.COM/a/b?x=y&c=d`" converts it to become "`com,example,foo)/a/b?c=d&x=y`", which changes the domain name with lower case letters, removes the "www" subdomain, reverses the order of hostname segments, and sorts query parameters. *SURTs* are commonly used in archival index files and many other places where a URI is used as a lookup field, including *MementoMap*. A visual representation of *SURT* is illustrated in Figure 1 in the form of a tree that segregates layers of *Scheme*, *Host*, *Path*, and *Query*. It further annotates various depths of *Host* and *Path* segments as *H0, H1, H2*… and *P0, P1, P2*… that will be useful in understanding some terminologies used later in this paper. *SURTs* also allow credentials and port numbers, but we omitted them from the illustration for brevity. It is worth noting that the scheme portion is common in all *HTTP/HTTPS* URIs and has no informational value, hence the "`https://(`" prefix is often omitted.

```
!context  ["https://git.io/mementomap"]
!id       {uri: "https://archive.example.org/"}
!fields   {keys: ["surt"], values: ["frequency"]}
!meta     {name: "Example Archive", year: 1996}
!meta     {type: "MementoMap"}
!meta     {updated_at: "2018-09-03T13:27:52Z"}
*                        54321/20000
com,*                    10000+
org,arxiv)/              100
org,arxiv)/*             2500~/900
org,arxiv)/pdf/*         0
uk,co,bbc)/images/* 300+/20-
```

Figure 2: A Sample MementoMap in UKVS Format

UKVS (Unified Key Value Store) [4] is an evolving file format proposal that is a contribution of this *MementoMap* work. It is an evolution of the *CDXJ* format that we earlier proposed to be used by *Archive Profiles* [9]. This format extends *SURT* with wildcard support and improves various other aspects to simplify it and eliminate some limitations of our prior proposal (such as not being able to express blacklists or lack of support to merge two profiles generated with different profiling policies). We generalized the format to be more inclusive and flexible after we realized its utility in many web archiving related use cases (such as indexing, replay access control list (ACL), fixity blocks [15], and extended *TimeMaps*) and many other places such as extended server logging.

Figure 2 illustrates a sample *MementoMap* file that starts with some metadata headers. Header lines are prefixed with "`!`" to ensure they are separated from data lines and surfaced on top when the file is sorted. The "`!fields`" header tells that the first column is a *SURT* and is used as a lookup key (there can be more than one key column such as *Datetime* or *Language*) which is followed by a value column that holds "`frequency`" information. Each data line can optionally also contain a single-line JSON block, which is not illustrated here for the sake of simplicity. The `frequency` column is formatted as "`[URI-M Count]/[URI-R Count]`" where both counts are optional and the separator is also optional if only the `URI-M`

[3]http://timetravel.mementoweb.org/
[4]https://github.com/iipc/openwayback
[5]https://github.com/webrecorder/pywb

Table 1: Top Arquivo.pt TLDs

TLD	Mementos
pt	3,360,790,396
com	967,044,865
eu	209,800,729
net	90,049,592
org	62,198,772
br	23,156,717
mz	20,395,256
de	16,889,341
cv	16,494,652
ao	14,546,495

Table 2: Arquivo.pt CDXJ Statistics

Attributes	Values
CDXJ files	70
Total file size	1.8T
Compressed file size	262G
Temporal coverage	1992–2018
CDXJ lines	5.0B
Mementos (URI-Ms)	4.9B
Unique URI-Rs	2.0B
Unique HxPx keys	1.1B
Unique hosts	5.8M
Unique IP addresses	15K

`Count` is present (in this paper we only used this latter option). Additionally, these counts can have an optional suffix character `+`, `-`, or `~` to express that the numbers are not exact and represent a lower bound, an upper bound, and a rough estimate respectively. The first data line in the example means there are a total of exactly 54,321 mementos (*URI-Ms*) of exactly 20,000 *URI-Rs* in the archive and the next line suggests that there are at least 10,000 mementos from the "`.com`" *TLD*. The next two lines suggest that there are 100 mementos of the `arxiv.org` homepage and many more captures of pages with deeper paths. However, the next line illustrates an exclusion of a sub-tree by being more specific under `/pdf/*` that has zero mementos (this is illustrated in Figure 1 as well).

Arquivo.pt [23] was founded in 2008 with the aim to preserve web content of interest to the Portuguese community, but not limited to just the `.pt` *TLD* (as shown in Table 1). It has since archived about 5B mementos of which some data was donated to it by other archives, including IA, explaining why its temporal spread extends back before the PWA's founding date. We analyzed 1.8T of PWA's complete *CDXJ* index in production. A brief summary of the dataset is shown in Table 2. We used it along with ODU's MemGator server logs to evaluate this work.

3 RELATED WORK

Query routing is a rigorously researched topic in various fields including, networked databases, meta-searching, and search aggregation [24, 31]. However, archive profiling and Memento lookup routing is a niche field that is not explored by many researchers beyond a small community.

Sanderson et al. created comprehensive content-based profiles [34, 35] of various *International Internet Preservation Consortium (IIPC)* member archives by collecting their *CDX* files and extracting URI-Rs from them. This approach gave them complete knowledge of the holdings in each participating archive, hence they can route queries precisely to archives that have any mementos for the given *URI-R*. This approach yielded no false positives or false negatives (i.e., 100% *Accuracy*) while the *CDX* files were fresh, but they would go stale very quickly. It is a resource and time intensive task to generate such profiles and some archives may be unwilling or unable to provide their CDX files. Such profiles are so large in size (typically, a few billion *URI-R* keys) that they require special infrastructure to support fast lookup. Acquiring fresh *CDX* files from various archives and updating these profiles regularly is not easy.

In contrast, AlSum et al. explored a minimal form of archive profiling using only the *TLDs* and *Content-Language* [12, 13]. They created profiles of 15 public archives using access logs of those archives (if available) and fulltext search queries. They found that by sending requests to only the top three archives matching the criteria for the lookup URI based on their profile, they can discover about 96% of *TimeMaps*. When they excluded IA from the list and performed the same experiment on the remaining archives, they were able to discover about 65% of *TimeMaps* using the remaining top three archives. Excluding IA was an important aspect of evaluation as its dominance can cause bias in results. This exclusion experiment also showed the importance of smaller archives and the impact of their holdings collectively. This minimal approach had many false positives, but no false negatives.

Bornand et al. implemented a different approach for Memento routing by building binary classifiers from LANL's Time Travel aggregator cache data [17]. They analyzed responses from various archives in the aggregator's cache over a period of time to learn about the holdings of different archives. They reported a 77% reduction in the number of requests and a 42% reduction in response time while maintaining 85% *Recall*. These approaches can be categorized as usage-based profiling in which access logs or caches are used to observe what people were looking for in archives and which of those lookups had a hit or miss in the past. While usage-based profiling can be useful for Memento lookup routing, it may not give the real picture of archives' holdings, producing both false negatives and false positives[6].

We found that people looked for less than 0.003% of the archived resources in PWA using MemGator service. There is a need for content-based archive profiling which can express what is present in archives, irrespective of whether or not it is being looked for.

In previous work [8, 10], we explored the middle ground where archive profiles are neither as minimal as storing just the *TLD* (which results in many false positives) nor as detailed as collecting every URI-R present in every archive (which goes stale very quickly and is difficult to maintain). We first defined various profiling policies, summarized *CDX* files according to those policies, evaluated associated costs and benefits, and prepared gold standard datasets [8, 10]. In our experiments, we correctly identified about 78% of the URIs that were or were not present in the archive with less than 1% relative cost as compared to the complete knowledge profile and identified 94% URIs with less than 10% relative cost without any false negatives. Based on the archive profiling framework we established, we further investigated the possibility of content-based profiling by issuing fulltext search queries (when available) and observing returned results [11] if access to the *CDX* data is not possible. We were able to make routing decisions of 80% of the requests correctly while maintaining about 90% *Recall* by discovering only 10% of the archive holdings and generating a profile that costs less than 1% of the complete knowledge profile. *MementoMap* is a continuation of this effort to make it more flexible and portable by eliminating the need for rigid profiling policies we defined earlier (which are still good for baseline evaluation purposes) and replacing them with an adaptive approach in which the level of detail is dynamically controlled with a number of parameters.

[6]https://groups.google.com/forum/#!topic/memento-dev/YE4rt6L5ICg

```
func host_keys(surt)
  s = surt.split(")")[0].split(",", MAXHOSTDEPTH)
  return [s[:i].join(",") for i in 1..len(s)]

func path_keys(surt)
  s = surt.split("?")[0].split("/", MAXPATHDEPTH)
  return [s[:i].join("/") for i in 1..len(s)]

func compact(imap, omap, opts)
  htrail = [None] * MAXHOSTDEPTH
  ptrail = [None] * MAXPATHDEPTH
  for line in imap
    key, freq, _ = line.split()
    k = host_keys(key)
    for i in range(len(k))
      if htrail[i] == k[i] # Existing branch
        htrail[i][1] += freq
      else # New branch
        for j in range(i, MAXHOSTDEPTH)
          if rollup_threshold_reached
            omap.seek(htrail[j][3]) # Move back
            omap.write(htrail[j][:1].join(",* "))
            reset_remaining_trail(ptrail, 0)
        reset_remaining_trail(htrail, i)
      if !htrail[i] # New tree node
        htrail[i] = [k[i], freq, 0, omap.tell()]
        htrail[i-1][2]++ # Incr parent's children
    # Repeat similar logic for path segment
    omap.write(line)
    omap.truncate() # Clear any rollup residue

func lookp_keys(uri)
  key = surtify(uri).split("?")[0].strip("/")
  keys = [key]
  while "," in key
    keys.uppend(sub("(.+[,/]).+$", "\1*", key))
    key = sub("(.+)[,/].+$", "\1", key)
  return keys

func lookup(mementomap, uri)
  for key in lookp_keys(uri)
    result = bin_search(mementomap, key)
    if result
      return [key, result]
```

Figure 3: MementoMap Compaction and Lookup Procedures

4 METHODOLOGY

Generating a *MementoMap* begins by scanning *CDX/CDXJ* files, performing fulltext search, filtering access logs, or any other means to identify what *URIs* an archive holds (or does not hold). These *URIs* are then converted to *SURTs* (if not already) and their query section is stripped off. We call these partial *SURTs* as *HxPx URI Keys* (which means a *URI Key* that has all the host and path parts, but no query parameters). We earlier found that removing query parameters from these *SURTs* reduces the file size and the number of unique *URI Keys* significantly without any significant loss in the lookup *Accuracy* [10]. We then create a text file with its first column containing *HxPx Keys* and the second column as their respective *Frequencies*. The *frequency* column in its simplest form can be the count of each *HxPx Key*, but it can be made more expressive as illustrated in the data section of Figure 2. Finally, necessary metadata is added and the file is sorted as the baseline *MementoMap*.

In order to make a less detailed *MementoMap* (which is desired for efficient dissemination and long-lasting freshness at the cost of increased false positives), we pass a detailed *MementoMap* through a compaction procedure which yields a summarized output that contains fewer lookup keys by rolling sub-trees with many children nodes up and replacing them with corresponding wildcard keys. Our compaction algorithm is illustrated with pseudo-code in Figure 3. As opposed to an in-memory tree building (which will not scale), it is a single-pass procedure with minimal memory requirements and does not need any special hardware to process a *MementoMap* of any size. We leverage the fact that the input *MementoMap* is sorted, hence, we can easily detect at what depth of host or path segments a branch differed from the previous line. We keep track of the most recent state of host and path keys at each depth (up to `MAXHOSTDEPTH` and `MAXPATHDEPTH`), their corresponding cumulative frequencies, how many children nodes each of them have seen so far, and the byte position of the output file when these keys were seen the first time. Each time we encounter a new branch at any depth, we check to see if a roll up action is applicable at that depth or further down in the existing tree based on the most recent states and the compaction parameters supplied. If so, we move the write pointer in the output file back to the position where the corresponding key was observed first, then we reset the state of all the deeper depths and update them with the current state. As a consequence of this progressive processing, the trailing part of the output file is overwritten many times. The input file does not have to be the baseline *MementoMap*, any *MementoMap* can be supplied as input with fresh compaction parameters to attempt to further compact it. Our algorithm is parallel processing-friendly if the input data is partitioned strategically (e.g., processing each *TLD*'s records on separate machines and combining all compacted output files). It is worth noting that sub-trees of the path section are neither independent trees nor have a single root node (as shown in Figure 1), as a result, certain implementation details can be more complex than a simple tree pruning algorithm.

The algorithm for lookup in a *MementoMap* is also illustrated in Figure 3. Given a *URI*, we first generate all possible lookup keys, in which all keys but the longest one have a wildcard suffix (e.g, "`Www.Example.COM/a/b?x=y&c=d`" yields "`com,example)/a/b`", "`com,example)/a/b/*`", "`com,example)/a/*`", "`com,example)/*`", and "`com,*`" as lookup keys). We then perform a binary search in the *MementoMap* with lookup keys in decreasing specificity until we find a match or all the keys are exhausted. In case of a match, we return the matched lookup key and corresponding frequency results.

5 EVALUATION

For evaluation we used the complete index of Arquivo.pt, complete logs of our MemGator service, and generated *MementoMaps*. We first examine logs, then describe holdings of PWA in detail, and finally measure the effectiveness of various *MementoMaps*.

5.1 Archived vs. Accessed Resources

We analyzed over three years of our *MemGator* logs containing records about 14 different web archives. In its lifetime it has served a total of 5,241,771 requests for 3,282,155 unique *URIs*. Table 3 shows the the summary of our log analysis in which IA has over 35% hit rate, and every other archive is below 10% (down to zero) in decreasing order of hit rate. Arquivo.pt is showing a 3.35% hit rate, so we cross checked it with the full index and found that there are only 1.64% unique *URIs* from the *MemGator* logs that are present in PWA (note that the CDX data even includes recent mementos that

Table 3: MemGator Log Responses from Various Archives from 2015/10/25 to 2019/01/16

Archive	Request	Hit%	Miss%	Err%	Sleep
Internet Archive	4,723,880	35.76	63.68	0.56	1,594
Archive-It	5,011,385	9.14	90.38	0.48	1,556
Archive Today	5,151,720	8.44	88.96	2.60	1,920
Library of Congress	4,862,458	4.77	94.31	0.92	2,705
Arquivo.pt	4,300,221	3.35	96.29	0.36	1,153
Icelandic	5,126,706	2.22	97.14	0.64	3,143
Stanford	5,178,835	1.54	98.02	0.43	1,482
UK Web Archive	5,113,984	1.49	86.30	12.20	2,779
Perma	4,116,099	1.32	98.67	0.01	46
PRONI	5,165,805	0.75	98.72	0.54	1,608
UK Parliament	5,181,991	0.63	98.85	0.52	1,542
NRS	2,683,311	0.21	99.77	0.01	46
UK National	5,178,184	0.10	99.45	0.45	1,457
PastPages	22,058	0.00	62.90	37.10	0
All	61,816,637	5.44	92.92	1.64	21,031

were not present earlier). The difference in these numbers is perhaps as a result of some archived *URIs* being looked for more frequently. This low percentage of overlap in access logs and archive indexes conforms to our earlier findings [10]. The table shows an overall 93% miss rate, which is all wasted traffic and delayed response time. Identifying sources of such a large miss rate can save resources and time significantly, which is the primary motivation of this work.

There are some other notable entries in Table 3 such as low number of requests to PastPages which was excluded from being polled in the early days due to its zero hit rate and high error rate. NRS (National Records of Scotland) is a new addition to the list, hence it shows a low number of requests. The high error rate of the UK Web Archive was primarily caused by a bug in the Go language (used to develop MemGator) that was not cleaning idle TCP connections that were already closed by the application. As a result, UKWA's firewall was seeing an ever increasing number of open, but idle connections, hence dropping packets after a hard limit of 20 concurrent connections per host. This has since been fixed after the release of the Go language version 1.7. We have later introduced an automatic dormant feature that puts an upstream archive to sleep for a configurable amount of time after a set number of successive errors.

Figure 4 shows a breakdown of what people are looking for in archives and what web archives hold. The 1.1K entry in the "Ones" row and "Tens" column shows that there are over a thousand *URI-Rs* that were requested 10–99 times in *MemGator* and each has 1–9 mementos in PWA. Large numbers in the "Zero" column show there are a lot of mementos that are never requested from *MemGator*. Similarly, the "Zero" row shows there are a lot of requests that have zero mementos in PWA. Another way to look at it is that a content-based archive profile will not know about the "Zero" row and a usage-based profile will miss out the content in the "Zero" column. Active archives may want to profile their access logs periodically to identify potential seed URIs of frequently requested missing resources that are within the scope of the archive. Ideally, we would like more activity along the diagonal that passes from the (Zero, Zero) corner, except, the corner itself, which suggests there are undetermined number of *URI-Rs* that were never archived or accessed.

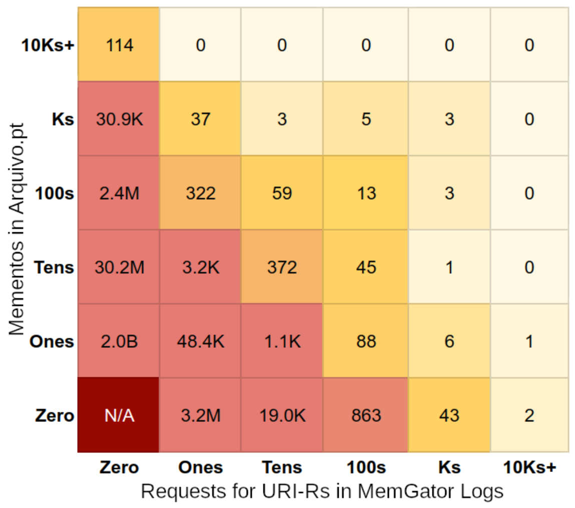
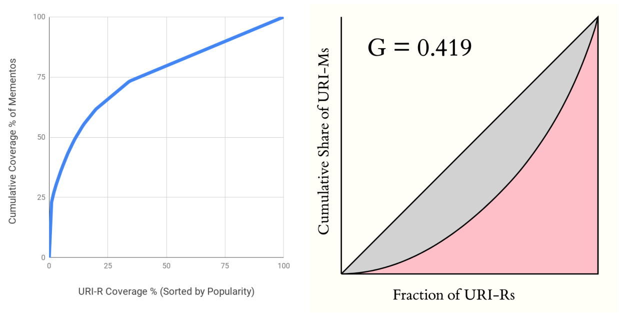


Figure 4: Overlap Between Archived and Accessed Resources

Table 4: URI-M vs. URI-R Summary of Arquivo.pt

Attributes	Values
Unique URI-Rs	1,999,790,376
Total number of mementos	4,923,080,506
Maximum mementos for any URI-R	2,308,634
Median (and Minimum)	1
Mean mementos per URI-R (γ)	2.46
Standard Deviation	57.20
Gini Coefficient	0.42
Pareto Break Point	70/30



(a) Percentage of URI-Rs by Popularity vs. Cumulative Percentage of Mementos

(b) Gini Coefficient of Memento Over URI-R Population

Figure 5: Distribution of Mementos Over URI-Rs in PWA

5.2 Holdings of Arquivo.pt

Table 4 and Figure 5 summarize the distribution of *URI-Ms* over *URI-Rs* in PWA. Almost 2M unique *URI-Rs* in PWA have an average of 2.46 mementos per *URI-R* (γ value [10]), but this distribution is not uniform. The top 30% *URI-Rs* account for 70% of the mementos, for a *Gini Coefficient* of 0.42 [40]. Additionally, the *Median* is one, which means at least half of the *URI-Rs* have only one memento.

Table 5: Most Archived URI-Rs in Arquivo.pt

URIs	URI-Ms
com,wunderground,icons)/graphics/blank.gif	2,308,634
com,wunderground,icons)/graphics/wuicorner.gif	768,250
pt,ipleiria,inscricoes)/logon.aspx	238,292
com,wunderground,icons)/graphics/wuicorner2.gif	207,448
com,lygo)/ly/i/inv/dot_clear.gif	115,221
com,listbot)/subscribe_button.gif	108,530
com,wunderground,icons)/* (including top URI-R)	3,336,086
com,wunderground,* (41 sub-domains)	3,392,676

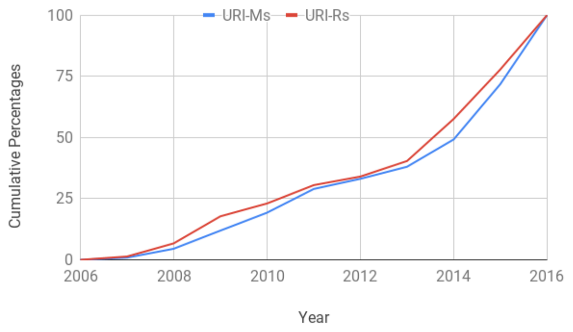


Figure 6: Cumulative Growth of URI-Rs and URI-Ms in PWA

Furthermore, the most frequently archived *URI-R* has 2.3M mementos (i.e., 0.05% of total), so we decided to investigate it further. Table 5 lists the six most archived *URI-Rs*, and they are mostly one pixel clear images and corner graphics primarily used in web designing in pre-CSS3 era. The only HTML page that shows up in the top list is a login page. We further investigated all the mementos from all the subdomains of the top *URI-R*'s domain and found that the `blank.gif` image was archived out of proportion. This shows another use for archive profiling – identifying such unintentional biases due to misconfigured crawling policies or bugs in crawlers' frontier queue management.

Furthermore, we partitioned PWA's index into yearly buckets for analysis as shown in Table 6. Data prior to year 2008 is mostly donated from other sources in the form of many small files, as PWA was not yet established. However, when everything is put together it looks like the archiving activity took off significantly in 2007. Low numbers in years 2017 and 2018 are due to PWA's embargo policy. It shows that PWA's collection is growing with a healthy pace by mostly collecting new *URI-Rs* as well as revisiting on an average 26% of older ones on a yearly basis. We expected γ would change gradually over time, but years 2000 and 2018 had significantly high values with respect to other years. So, we looked for the possibility of increased 3xx status codes in those years as a potential source of increase in γ (e.g., `http` *URIs* redirecting to corresponding `https` version), but we did not see any correlation there. However, the data for these years seems to have come from another source and overall they are insignificant, hence, the cumulative γ^+ is fairly stable between 2 and 3. We noted a significant and steady growth in 4xx status codes which has crossed the 20% mark in year 2016. Status codes for the last two years (still in embargo period) do not sum up to 100% because a significant portion of their entries are either *revisit* records or screenshots that do not report status codes. In Figure 6 we plotted a cumulative growth graph of both *URI-Ms* and *URI-Rs* to see the shape [27] of PWA during the active region. Their archiving rate is increasing over time as almost half of the total mementos were archived in the last two active years alone.

5.3 The Shape of Archived URI Tree

To understand the shape of the *URI Keys* tree in *MementoMap* we first investigated the number of unique *Domains* and *HxPx Keys* that have certain host or path depths as shown in Table 7. These numbers are relative to the size of the PWA index, but we believe a similar trend should be seen in other archives, unless their collection is manually curated and crawled using a more or less capable tool than what is currently being used by many large web archives [32]. There were some outliers in the data that showed a host depth of up to 15 and path depths up to 130, but those were very few in number. These numbers gave us a good starting point to decide how deep we need to analyze hosts and paths for profiling.

Table 8 is based on the total 1,138,923,169 unique *HxPx Keys* of PWA's current index. For example, the *H3* (see Figure 1 for naming convention) row means there are a total of 2,158,880 unique *H3* prefixes that cover a sum of 630,309,184 *HxPx Keys* of which the most popular prefix covers 51,849,377 keys alone. The *Mean* number of keys per prefix at *H3* is 291.96 with a median 7 and standard deviation 37,641.59. The *RedQ (Reduction Coefficient)* column represents a derived quantity that we defined as the amount of reduction in keys it would cause if *HxPx Keys* longer than a given depth are stripped off at that depth and only counted reduced unique prefixes. This can be calculated using Equation 1 at depth d where $|\text{HxPx Keys}_{\geq d}|$ is the number of *HxPx Keys* with depth $\geq d$ and $|\text{URI Keys}_d|$ is the number of unique partial *URI Keys* stripped at depth d (reported under the *Sum* and *Count* columns of Table 8 respectively). Figures 7(a) and 7(b) show the cumulative reduction as the top most frequent keys are rolled up at a host and path depth respectively. Furthermore, there are 253,091 nodes in the tree one depth above (i.e., *H2*) that lead to 2,158,880 nodes at the current depth. While the *Mean Child* count at *H3* is 8.53, the distribution is not uniform. Figures 7(c) and 7(d) show the cumulative reduction in immediate children count as the most popular parents leading to the current depth are rolled up incrementally from bottom up. The purpose of Reduction Coefficient is to understand the impact and importance of various host and path depths globally while the *Mean Child* count gives an estimate of a more localized impact at a given depth. For this work we have used the latter as a factor to decide when to roll a sub-tree up while compacting a *MementoMap*. Rolling the sub-tree up at *H1*, *H2*, and *P0* are not applicable for evaluation here because *H1* means shrinking everything into a single record of "*" key, *H2* would require out-of-band information (because not every *TLD* is equally popular), and *P0* being the root of the path has nothing to roll up into (though compaction might happen in the relevant host segment independently). We fit the remaining values of *Mean Child* count on Power Law [19] curves (other curve fittings are also possible) for both host and path segments to find a and k parameters and use these empirical values for compaction decision making.

Table 6: Yearly Distribution of URI-Rs, URI-Ms, and Status Codes in Arquivo.pt

Year	URI-R	URI-R$^+$	URI-M	URI-M$^+$	Dup. URI-R%	γ	γ^+	2xx%	3xx%	4xx%	5xx%
1992	1	1	1	1	0.00	1.00	1.00	100.00	0.00	0.00	0.00
1993	1	2	1	2	0.00	1.00	1.00	100.00	0.00	0.00	0.00
1994	128	130	225	227	0.00	1.76	1.75	100.00	0.00	0.00	0.00
1995	642	772	742	969	0.00	1.16	1.26	100.00	0.00	0.00	0.00
1996	110,531	111,303	126,600	127,569	0.00	1.15	1.15	99.96	0.01	0.00	0.00
1997	466,515	563,734	847,783	975,352	3.02	1.82	1.73	100.00	0.00	0.00	0.00
1998	447,042	928,112	747,114	1,722,466	18.49	1.67	1.86	99.23	0.77	0.00	0.00
1999	732,866	1,513,381	1,233,994	2,956,460	20.14	1.68	1.95	76.52	10.61	12.84	0.00
2000	1,710,099	2,874,152	13,413,518	16,369,978	20.43	7.84	5.70	86.99	7.24	5.73	0.00
2001	4,837,012	7,286,174	7,873,642	24,243,620	8.79	1.63	3.33	93.87	4.87	1.25	0.01
2002	7,675,876	13,364,488	13,048,749	37,292,369	20.81	1.70	2.79	90.96	5.11	3.92	0.01
2003	11,043,675	21,565,730	19,989,725	57,282,094	25.74	1.81	2.66	92.12	4.45	3.41	0.03
2004	11,550,512	29,460,627	22,810,763	80,092,857	31.65	1.97	2.72	92.00	5.11	2.88	0.01
2005	9,057,866	35,249,604	19,839,405	99,932,262	36.09	2.19	2.83	93.99	3.94	2.07	0.01
2006	5,979,310	39,609,628	15,388,836	115,321,098	27.08	2.57	2.91	92.33	6.29	1.37	0.01
2007	26,841,427	63,396,199	43,021,527	158,342,625	11.38	1.60	2.50	83.03	14.88	2.08	0.01
2008	113,915,969	166,926,098	174,996,303	333,338,928	9.12	1.54	2.00	85.87	8.95	6.18	0.37
2009	249,069,391	383,960,128	355,833,394	689,172,322	12.86	1.43	1.79	87.37	6.55	6.49	0.36
2010	174,786,328	487,044,797	352,019,433	1,041,191,755	41.02	2.01	2.14	87.39	6.83	6.49	0.42
2011	206,966,813	634,061,322	465,274,765	1,506,466,520	28.97	2.25	2.38	89.13	6.21	6.99	0.58
2012	118,916,669	703,235,309	200,042,923	1,706,509,443	41.83	1.68	2.43	87.79	6.66	7.96	0.46
2013	174,913,693	827,924,633	236,583,969	1,943,093,412	28.71	1.35	2.35	84.03	7.28	10.90	0.57
2014	430,555,712	1,166,054,663	536,560,181	2,479,653,593	21.47	1.25	2.13	80.50	7.10	13.47	0.52
2015	558,504,002	1,563,688,006	1,087,680,516	3,567,334,109	28.80	1.95	2.28	78.32	5.12	17.75	0.32
2016	719,889,903	1,999,522,571	1,353,786,928	4,921,121,037	39.46	1.88	2.46	73.20	6.46	20.78	1.30
2017	685,097	1,999,687,103	1,111,999	4,922,233,036	75.98	1.62	2.46	57.82	5.44	7.89	0.22
2018	106,186	1,999,790,376	847,470	4,923,080,506	2.74	7.98	2.46	22.07	5.63	1.38	0.00
All	1,999,790,376	1,999,790,376	4,923,080,506	4,923,080,506	0.00	2.46	2.46	80.74	6.42	13.86	0.66

Table 7: Unique Items with Exact Host and Path Depths

Depth	Host (Domains)	Host (HxPx)	Path (HxPx)
0	1	1	4,456,831
1	119	6,479	113,022,403
2	1,949,845	508,607,506	225,489,773
3	2,097,254	429,000,297	334,455,187
4	1,316,005	161,912,251	174,429,887
5	234,110	21,825,084	127,484,179
6	95,492	7,935,125	68,578,693
7	28,121	3,252,943	45,819,300
8	64,716	3,722,893	22,178,800
9	55,801	2,660,529	15,553,102
10	5	50	6,596,158
11+	5	12	858,856
Total	5,841,473	1,138,923,169	1,138,923,169

$$\text{RedQ}_d = \frac{|\text{HxPx Keys}_{\geq d}| - |\text{URI Keys}_d|}{|\text{HxPx Keys}|} \quad (1)$$

5.4 MementoMap Cost and Accuracy

Web archives are messy collections that contain many malformed records often caused by configuration issues in web servers, poorly written web applications, bugs in archiving tools, incompatible file transformations, or even security vulnerabilities [5]. Archive profiling can uncover some of these as we found many malformed *MIME-Type*[7] and *Status Code*[8] entries in PWA.

To run our experiments we decided to filter only the clean records out from these *CDXJ* files. We further limited our scope to only *HTML* pages that returned a `200` status code. Additionally, we excluded any `robots.txt` and `sitemap.xml` files that were served wrongly as "`text/html`". With these filters in place we reduced mementos by almost half of the total index size to only 2,671,653,766. Now, there are 962,832,513 filtered unique *URI-Rs*, which means the γ value is increased slightly to 2.77. Also, the *HxPx Keys* count is reduced to 447,107,301, which is 39% of the overall number. From these keys we created the baseline *MementoMap* with compressed file size of 3.4G which is already reduced to 1.3% of the original index size.

In the next step we supplied this baseline *MementoMap* as input for compaction with host and path compaction weights $W_h = 4.00$ and $W_p = 4.0$ respectively. These weights are multiplied by their corresponding estimated *Mean Child* value at each depth to find the cutoff number when the sub-tree is to be rolled up. A small weight will roll the sub-tree up more aggressively than a large value, resulting in a more compact *MementoMap*. This process produced a *MementoMap* with only 27,010,037 lines (i.e., 6.0% of the baseline or 2.8% *Relative Cost*, which is the ratio of reduced number of unique lookup keys vs. number of unique *URI-Rs* [9]) after going through

[7]https://gist.github.com/ibnesayeed/bb167fe19c5719d87c1c1f665001d44b
[8]https://gist.github.com/ibnesayeed/7307f0bf1783357db99f8b2357249dd0

Table 8: Host and Path Depth Statistics of Unique HxPx Keys in Arquivo.pt

Depth	Count	Sum	Max	Mean	Med.	StdDev	RedQ	Parents	Children	MeanChld
H1	973	1,138,923,169	616,372,626	1,170,527.41	930	21,620,107.00	1.00000	1	973	973.00
H2	2,068,333	1,138,916,690	109,176,956	550.64	5	91,308.66	0.99818	904	2,068,333	2,287.98
H3	2,158,880	630,309,184	51,849,377	291.96	7	37,641.59	0.55153	253,091	2,158,880	8.53
H4	1,329,137	201,308,887	3,765,122	151.46	10	4,797.10	0.17559	148,589	1,329,137	8.95
H5	245,881	39,396,636	376,969	160.23	5	3,420.96	0.03438	31,635	245,881	7.77
H6	103,579	17,571,552	105,591	169.64	27	1,106.03	0.01534	16,496	103,579	6.28
H7	34,380	9,636,427	19,572	280.29	20	450.16	0.00843	10,061	34,380	3.42
H8	69,829	6,383,484	535	91.42	120	45.75	0.00554	15,359	69,829	4.55
H9	55,811	2,660,591	80	47.67	56	19.6	0.00229	55,811	55,811	1.00
H10+	10	62	19	6.20	2	6.51	0.00000	10	10	1.00
P0	5,841,503	1,138,923,169	2,264,623	194.97	7	3,059.43	0.99487	5,841,503	5,841,503	1.00
P1	145,687,459	1,134,466,338	2,242,344	7.79	1	376.64	0.86817	5,828,059	145,687,459	25.00
P2	290,761,965	1,021,443,935	603,840	3.51	1	130.76	0.64156	40,130,355	290,761,965	7.25
P3	392,635,328	795,954,162	565,043	2.03	1	78.14	0.35412	79,234,027	392,635,328	4.96
P4	215,251,988	461,498,975	512,098	2.14	1	80.01	0.21621	66,059,544	215,251,988	3.26
P5	158,256,277	287,069,088	512,098	1.81	1	65.72	0.11310	48,163,114	158,256,277	3.29
P6	91,334,214	159,584,909	50,384	1.75	1	22.3	0.05993	33,776,599	91,334,214	2.70
P7	60,099,825	91,006,216	44,114	1.51	1	17.24	0.02714	24,201,781	60,099,825	2.48
P8	31,101,768	45,186,916	24,631	1.45	1	15.54	0.01237	14,890,308	31,101,768	2.09
P9	18,601,197	23,008,116	10,247	1.24	1	9.74	0.00387	9,233,634	18,601,197	2.01
P10	6,817,122	7,455,014	5,858	1.09	1	9.36	0.00056	3,206,260	6,817,122	2.13
P11+	858,772	858,856	2	1.00	1	0.01	0.00000	222,432	392,565	1.76

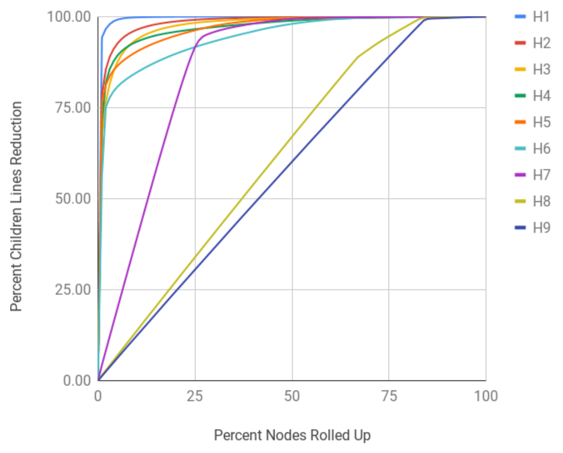
4,574,305 recursive roll ups. The process took 2.4 hours to complete on our Network File System (NFS) storage. The time taken to complete the compaction process is a function of the number of lines to process from the input, number of lines to be written out, and the number of roll ups to occur (along with the read and write speeds of the disk). Since the process is I/O intensive, using faster storage can reduce the time significantly, which we verified by repeating the experiment on *TMPFS* [38]. We generated 36 variations of *MementoMaps* with all possible pairs of W_h and W_p weights from values 4.00, 2.00, 1.00, 0.50, 0.25, and 0.00. To generate *MementoMaps* with smaller weights we used *MementoMaps* of immediate larger weight pairs as inputs (e.g., input one with $W_h = 2.00, W_p = 0.50$ to generate one with $W_h = 1.00, W_p = 0.25$). This technique of chaining the output as input to the next step reduced the generation time for subsequent *MementoMaps* from hours to a few minutes and also illustrated that *MementoMaps* can easily be compacted further when needed.

Figure 8 shows a portion of the roll up activity during the compaction process. The size of the output grows linearly, but on a micro-scale whenever there is a roll up activity, the output size goes down depending on at what depth roll up happened and how big of a sub-tree was affected.

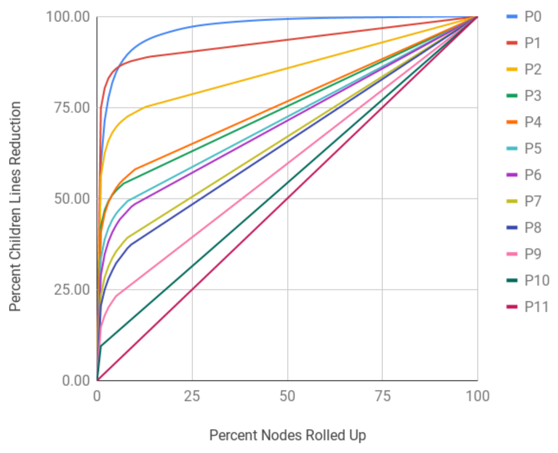
Finally, we used MemGator logs to perform lookup in these 36 *MementoMaps* generated with different host and path weight pairs to see how well they perform. Figure 9 shows the *Relative Cost* and corresponding *Lookup Routing Accuracy* of these *MementoMaps*. The *Accuracy* here is defined as the ratio of correctly identified *URIs* for their presence or absence vs. all the lookup *URIs*. In this experiment *MementoMaps* with weights $W_h = 4.00, W_p = 2.00$ and $W_h = 2.00, W_p = 2.00$ yielded about 60% *Routing Accuracy* with less 1.5% *Relative Cost* without any false negatives (i.e., 100% *Recall*). Since PWA had only 3.35% hit rate in the past three years, *MemGator* could have avoided almost 60% of the wasted traffic to PWA without missing any good results if PWA were to advertise its holdings via a small *MementoMap* of about 111MB in size. The accuracy can further be improved by 1) exploring other optimal configurations for sub-tree pruning, 2) generating *MementoMaps* with the full index, not just a sample, and 3) including entries for absent resources from the “Zero” row of the Figure 4.

6 CONCLUSIONS AND FUTURE WORK

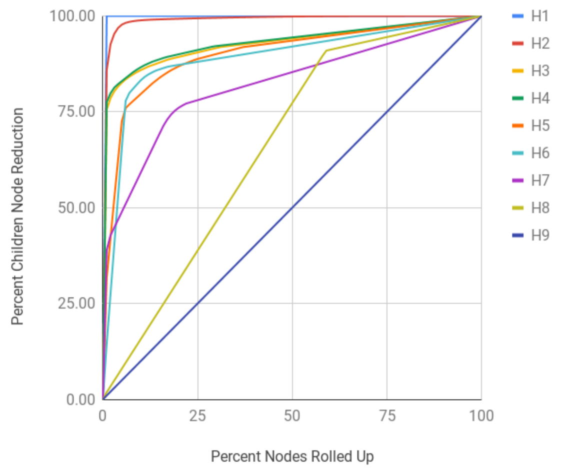
In this work we proposed *MementoMap*, a flexible and adaptive framework to express holdings of a web archive efficiently. We described a simple, yet extensible, file format suitable for *MementoMap* and some other use cases. We extended traditional *SURT* format to support wildcards for partial *URI Keys*. We analyzed more than three years of *MemGator* logs to understand the response behavior of 14 public web archives. We used the complete index of 5B mementos in the Arquivo.pt as a case study, learned some generalizable behaviors of *URIs* in web archives, described Arquivo.pt’s holdings in different ways, and created *MementoMaps* of varying sizes from it for evaluation. We designed a single-pass, memory-efficient, and parallelization-friendly algorithm to compact a large *MementoMap* into a small one iteratively, based on user-specified parameters to accommodate different needs and available resources. We also implemented a time-and memory-efficient lookup method using binary search on *MementoMap* files on disk by leveraging the fact that *MementoMaps* are in a lexicographical order. Finally, we evaluated the effectiveness of *MementoMaps* of varying sizes by measuring the *Accuracy* using 3.3M unique *URIs* from *MemGator* logs. We found that a *MementoMap* of less than 1.5% *Relative Cost* can correctly identify the presence or absence of 60% of the lookup *URIs* in the corresponding archive without any false negatives. We open-sourced our implementation code under a permissive license [3].



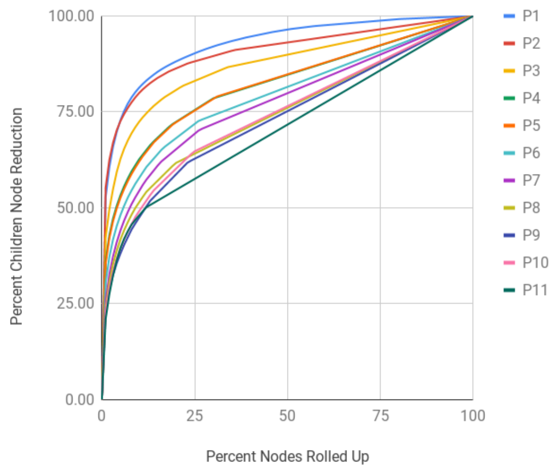
(a) Global HxPx Reduction Rate at Host



(b) Global HxPx Reduction Rate at Path



(c) Cumulative Host Children Reduction



(d) Cumulative Path Children Reduction

Figure 7: Global and Incremental Host and Path Segment Reduction

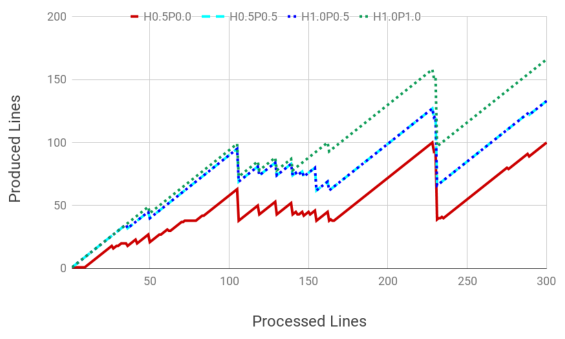


Figure 8: Growth of Compacted MementoMap vs. Lines Processed from an Input MementoMap

The trend shown in Figure 7 opens up many possibilities to try, such as, to fit them as Heaps' Law [21] curves and estimate K and β parameters to then automatically identify the best roll up possibilities instead of asking a human to provide weights and supply other parameters. The *MementoMap* format proposed in this paper supports the ability to highlight inactive sub-trees within an active tree by being more specific, which will reduce false positives. However, generating this information will require processing access logs or other out-of-band data sources. Rolling the sub-tree up at *H2* can be useful for large web archives and one way to explore this possibility is to identify globally less popular *TLDs* that have a significant presence in an archive. Currently, it is possible to do it manually, but not automatically. A major goal of this work is to push for adoption of *MementoMap* by adding out-of-the-box support in major archival replay systems. We would also like to investigate the possibility

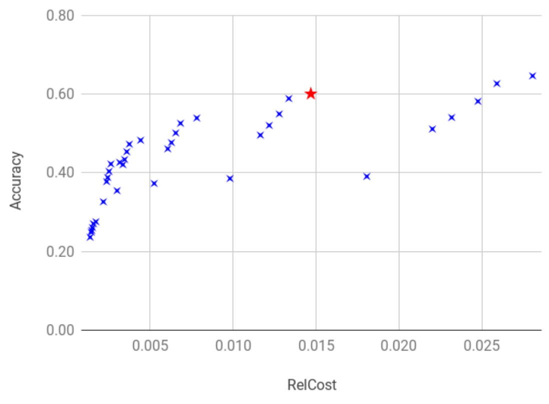


Figure 9: Relative Cost vs. Lookup Routing Accuracy

of routing non-*HTML* lookup requests by utilizing *MementoMap* generated for *HTML* mementos only. The motivation comes from the assumption that page requisites are generally co-located with the parent page, hence we can leverage the information present in the *Referer* header of embedded resources to identify potential archives to poll from.

7 ACKNOWLEDGEMENTS

Supported in part by NSF grant IIS-1526700.

REFERENCES

[1] Reem Al-Masri and James Cain. 2017. In Jordan, the "Invisible Hand" Blocks Internet Archive. https://www.7iber.com/technology/the-invisible-hand-blocks-internet-archive/.

[2] Sawood Alam. 2015. MemGator: A Memento Aggregator CLI and Server in Go. https://github.com/oduwsdl/MemGator.

[3] Sawood Alam. 2019. MementoMap: A Tool to Summarize Web Archive Holdings. https://github.com/oduwsdl/MementoMap.

[4] Sawood Alam. 2019. Unified Key Value Store (UKVS). https://github.com/oduwsdl/ORS/blob/master/ukvs.md.

[5] Sawood Alam, Charles L. Cartledge, and Michael L. Nelson. 2014. *Support for Various HTTP Methods on the Web*. Technical Report arXiv:1405.2330.

[6] Sawood Alam, Ilya Kreymer, and Michael L. Nelson. 2015. Object Resource Stream (ORS) and CDX-JSON (CDXJ) Draft. https://github.com/oduwsdl/ORS.

[7] Sawood Alam and Michael L. Nelson. 2016. MemGator - A Portable Concurrent Memento Aggregator. In *Proceedings of the 16th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '16)*.

[8] Sawood Alam, Michael L. Nelson, Herbert Van de Sompel, Lyudmila L. Balakireva, Harihar Shankar, and David S. H. Rosenthal. 2015. Web Archive Profiling Through CDX Summarization. In *Proceedings of 19th International Conference on Theory and Practice of Digital Libraries, TPDL 2015*. 3–14.

[9] Sawood Alam, Michael L. Nelson, Herbert Van de Sompel, Lyudmila L. Balakireva, Harihar Shankar, and David S. H. Rosenthal. 2016. Web Archive Profiling Through CDX Summarization. *International Journal on Digital Libraries* 17, 3 (2016), 223–238. https://doi.org/10.1007/s00799-016-0184-4

[10] Sawood Alam, Michael L. Nelson, Herbert Van de Sompel, Lyudmila L. Balakireva, Harihar Shankar, and David S. H. Rosenthal. 2016. Web Archive Profiling Through CDX Summarization. *International Journal on Digital Libraries* 17, 3 (2016), 223–238. https://doi.org/10.1007/s00799-016-0184-4

[11] Sawood Alam, Michael L. Nelson, Herbert Van de Sompel, and David S. H. Rosenthal. 2016. Web Archive Profiling Through Fulltext Search. In *Proceedings of 20th International Conference on Theory and Practice of Digital Libraries, TPDL 2016*. 121–132.

[12] Ahmed AlSum, Michele C. Weigle, Michael L. Nelson, and Herbert Van de Sompel. 2013. Profiling Web Archive Coverage for Top-Level Domain and Content Language. In *Proceedings of the International Conference on Theory and Practice of Digital Libraries, TPDL 2013*. 60–71.

[13] Ahmed AlSum, Michele C. Weigle, Michael L. Nelson, and Herbert Van de Sompel. 2014. Profiling Web Archive Coverage for Top-Level Domain and Content Language. *International Journal on Digital Libraries* 14, 3-4 (2014), 149–166.

[14] Grant Atkins. 2017. Carbon Dating the Web, version 4.0. http://ws-dl.blogspot.com/2017/09/2017-09-19-carbon-dating-web-version-40.html.

[15] Mohamed Aturban, Sawood Alam, Michael L. Nelson, and Michele C. Weigle. 2019. Archive Assisted Archival Fixity Verification Framework. In *Proceedings of the 19th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '19)*.

[16] John A. Berlin, Mat Kelly, Michael L. Nelson, and Michele C. Weigle. 2017. WAIL: Collection-Based Personal Web Archiving. In *Proceedings of the IEEE/ACM Joint Conference on Digital Libraries (JCDL)*. 340–341. https://doi.org/10.1109/JCDL.2017.7991619

[17] Nicolas Bornand, Lyudmila Balakireva, and Herbert Van de Sompel. 2016. Routing Memento Requests Using Binary Classifiers. In *Proceedings of the 16th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '16)*. ACM, 63–72.

[18] Chris Butler. 2017. Statement and Questions Regarding an Indian Court's Order to Block archive.org. https://blog.archive.org/2017/08/09/statement-and-questions-regarding-an-indian-courts-order-to-block-archive-org/.

[19] Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. 2009. Power-Law Distributions in Empirical Data. *SIAM review* 51, 4 (2009), 661–703. https://doi.org/10.1137/070710111

[20] Douglas Crockford. 2006. The application/json Media Type for JavaScript Object Notation (JSON). RFC 4627.

[21] Leo Egghe. 2007. Untangling Herdan's Law and Heaps' Law: Mathematical and Informetric Arguments. *Journal of the American Society for Information Science and Technology* 58, 5 (2007), 702–709.

[22] Adam Clark Estes. 2015. Russia Is Banning the Internet Archive and Blaming It On Terrorism. https://gizmodo.com/russia-is-banning-the-internet-archive-and-blaming-it-o-1713926987.

[23] Daniel Gomes, Miguel Costa, David Cruz, João Miranda, and Simão Fontes. 2013. Creating a Billion-Scale Searchable Web Archive. In *Proceedings of the 22nd International Conference on World Wide Web*. 1059–1066.

[24] Luis Gravano, Chen-Chuan K. Chang, Héctor García-Molina, and Andreas Paepcke. 1997. STARTS: Stanford Proposal for Internet Meta-Searching. *SIGMOD Record* 26, 2 (1997), 207–218. https://doi.org/10.1145/253262.253299

[25] GreatFire.org. 2018. www.archive.org is 100% blocked in China. http://archive.is/JdTnl.

[26] Internet Archive. 2003. CDX File Format. http://archive.org/web/researcher/cdx_file_format.php.

[27] Shawn M. Jones, Alexander Nwala, Michele C. Weigle, and Michael L. Nelson. 2018. The Many Shapes of Archive-It. In *Proceedings of iPres 2018*.

[28] Brewster Kahle. 2016. Geez, Now Internet Insurance? https://blog.archive.org/2016/06/16/geez-now-internet-insurance/.

[29] Mat Kelly, Sawood Alam, Michael L. Nelson, and Michele C. Weigle. 2016. InterPlanetary Wayback: Peer-To-Peer Permanence of Web Archives. In *Proceedings of the 20th International Conference on Theory and Practice of Digital Libraries*. 411–416. https://doi.org/10.1007/978-3-319-43997-6_35

[30] Mat Kelly, Michael L. Nelson, and Michele C. Weigle. 2014. Mink: Integrating the Live and Archived Web Viewing Experience Using Web Browsers and Memento. In *Proceedings of the 14th ACM/IEEE-CS Joint Conference on Digital Libraries*. 469–470.

[31] Weiyi Meng, Clement Yu, and King-Lup Liu. 2002. Building Efficient and Effective Metasearch Engines. *ACM Computing Surveys (CSUR)* 34, 1 (2002), 48–89.

[32] Gordon Mohr, Michael Stack, Igor Rnitovic, Dan Avery, and Michele Kimpton. 2004. Introduction to heritrix. In *Proceedings of the 4th International Web Archiving Workshop*.

[33] Alexander C. Nwala. 2015. I Can Haz Memento. http://ws-dl.blogspot.com/2015/07/2015-07-22-i-can-haz-memento.html.

[34] Robert Sanderson. 2012. Global Web Archive Integration with Memento. In *Proceedings of the 12th ACM/IEEE Joint Conference on Digital Libraries*. 379–380.

[35] Robert Sanderson, Herbert Van de Sompel, and Michael L. Nelson. 2012. IIPC Memento Aggregator Experiment. http://www.netpreserve.org/sites/default/files/resources/Sanderson.pdf.

[36] Kristinn Sigurðsson, Michael Stack, and Igor Ranitovic. 2006. Heritrix User Manual: Sort-friendly URI Reordering Transform. http://crawler.archive.org/articles/user_manual/glossary.html#surt.

[37] Erika Siregar. 2017. Deploying the Memento-Damage Service. http://ws-dl.blogspot.com/2017/11/2017-11-22-deploying-memento-damage.html.

[38] Peter Snyder. 1990. tmpfs: A Virtual Memory File System. In *Proceedings of the Autumn 1990 EUUG Conference*. 241–248.

[39] Herbert Van de Sompel, Michael L. Nelson, and Robert Sanderson. 2013. HTTP Framework for Time-Based Access to Resource States – Memento, Internet RFC 7089. https://tools.ietf.org/html/rfc7089.

[40] Shlomo Yitzhaki. 1979. Relative Deprivation and the Gini Coefficient. *The Quarterly Journal of Economics* (1979), 321–324.