Web Archive Profiling Through CDX Summarization

Sawood Alam[1], Michael L. Nelson[1], Herbert Van de Sompel[2], Lyudmila L. Balakireva[2], Harihar Shankar[2], and David S. H. Rosenthal[3]

[1] Computer Science Department, Old Dominion University, Norfolk, VA (USA)
{salam,mln}@cs.odu.edu
[2] Los Alamos National Laboratory, Los Alamos, NM (USA)
{herbertv,ludab,harihar}@lanl.gov
[3] Stanford University Libraries, Stanford, CA (USA)
dshr@stanford.edu

Abstract. With the proliferation of public web archives, it is becoming more important to better profile their contents, both to understand their immense holdings as well as support routing of requests in the Memento aggregator. To save time, the Memento aggregator should only poll the archives that are likely to have a copy of the requested URI. Using the CDX files produced after crawling, we can generate profiles of the archives that summarize their holdings and can be used to inform routing of the Memento aggregator's URI requests. Previous work in profiling ranged from using full URIs (no false positives, but with large profiles) to using only top-level domains (TLDs) (smaller profiles, but with many false positives). This work explores strategies in between these two extremes. In our experiments, we gained up to 22% routing precision with less than 5% relative cost as compared to the complete knowledge profile without any false negatives. With respect to the *TLD-only* profile, the registered domain profile doubled the routing precision, while complete hostname and one path segment gave a five fold increase in routing precision.

Keywords: Web Archives, Profiling, CDX Files, Memento

1 Introduction

The number of public web archives supporting the Memento protocol [17] natively or through proxies continues to grow. The Memento Aggregator [12], the Time Travel Service[1], and other services, both research and production, need to know which archives to poll when a request for an archived version of a file is received. In previous work, we showed that simple rules are insufficient to accurately model a web archive's holdings [4, 3]. For example, simply routing requests for `*.uk` URIs to the UK National Archives is insufficient: many other

[1] `http://timetravel.mementoweb.org/`

archives hold `*.uk` URIs, and the UK National Archives holds much more than just `*.uk` URIs. This is true for the many other national web archives as well.

In this paper we examine strategies for producing *profiles* of web archives. The idea is that profiles are a light-weight description of an archive's holdings to support applications such as coordinated crawling between archives, visualization of the archive's holdings, or routing of requests to the Memento Aggregator. It is the latter application that is the focus of this paper.

An archive profile has an inherent trade-off in its size vs. its ability to accurately describe the holdings of the archive. If a profile records each individual original URI (URI-R in Memento terminology) the size of the profile can grow quite large and difficult to share, query, and update. On the other hand, an aggregator making routing decisions will have perfect knowledge about whether or not an archive holds archived copies of the page, or mementos (URI-Ms in Memento terminology). On the other hand, if a profile contains just the summaries of top-level domains (TLDs) of an archive the profile size will be small but can result in many unnecessary queries being sent to the archive. For example, the presence of a single memento of `bbc.co.uk` will result in the profile advertising `.uk` holdings even though this may not be reflective of the archive's collection policy.

In this paper we examine various policies for generating profiles, from the extremes of using the entire URI-R to just the TLD. Using the CDX files[2] of the UK Web Archive (covering 10 years and 0.5 TB) and the ODU copy of the Archive-It (covering 14 years and 1.8 TB), we examine the trade-offs in profile size and routing precision for three million URIs requests.

2 Related Work

Query routing is common practice in various fields including meta-searching and search aggregation. Memento query routing was explored in the two efforts described below, but they explored extreme cases of profiling. We believe that an intermediate approach that gives flexibility with regards to balancing accuracy and effort can result in better and more effective routing.

Sanderson et al. created exhaustive profiles [13] of various IIPC member archives by collecting their CDX files and extracting URI-Rs from them (we denote it as *URIR Profile* in this paper). 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 (URI-M) for the given URI-R. 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 big 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.

[2] CDX files are created as an index of the WARC [10] files generated from the Heritrix web crawler; see [8] for a description of the CDX file format.

Many web archives tend to limit their crawling and holdings to some specific TLDs, for example, the British Library Web Archive prefers sites with `.uk` TLD. AlSum et al. created profiles based on TLD [4, 3] in which they recorded *URI-R Count* and *URI-M Count* under each TLD for twelve public web archives. Their results show that they were able to retrieve the complete TimeMap [17] in 84% of the cases using only the top 3 archives and in 91% of the cases when using the top 6 archives. This simple approach can reduce the number of queries generated by a Memento aggregator significantly with some loss in coverage.

3 Methodology

In this study we used CDX files to generate profiles, but profile generation is not limited to only CDX processing, it can also be done by sampling URI sets and querying the live archives or by using full-text searching feature provided by some archives. To deal with periodic updates of profiles, smaller profiles are generated with new data and these small profiles are merged into the base profile. Without an option to merge smaller profiles to build a large profile gradually, updates will require a complete reprocessing of the entire dataset, including the dataset previously processed. In these two cases the statistical measures such as URI-R count cannot have absolute values, so we use the sum of URI-M counts (as "frequency") from all the profiles under each *URI-Key* and keep track of the number of profiles they came from (as "spread") as an indication of the holdings as shown in Figure 1.

URI-Key is a term we introduced to describe the keys generated from a URI based on various policies. So far we have created policies that can be classified in two categories, *HmPn* and *DLim*. Policies of the generic form *HmPn* mean that the keys will have a maximum of "m" segments from the hostname and a maximum of "n" segments from the path. A *URI-Key* policy with only one hostname segment and no path segments (*H1P0*) is called *TLD-only* policy (as discussed in Section 2). *H3P0* policy covers most of the registered domains (that have one or two segments in their suffix [11], such as `.com` or `.co.uk`). If the number of segments are not limited, they are denoted with an "x", for example, *HxP1* policy covers any number of hostname segments with maximum of one path segment and *HxPx* means any number of hostname and path segments. Note that the *HxPx* policy is not the same as the *URIR* policy (as discussed in Section 2) because it strips off the query parameters from the URI, while the *URIR* policy stores complete URIs. Policies of the generic form *DLim* are based on the registered domain name, the number of segments in sub-domain, path, and query sections of a URI and the initial letter

```
1 @context https://oduwsdl.github.io/contexts/archiveprofile.jsonld
2 @id http://www.webarchive.org.uk/ukwa/
3 @about {"name": "UKWA 1996 Collection", "type": "urikey#H3P1", "...": "..."}
4 com,dilos)/region {"frequency": 14, "spread": 2}
5 edu,orst)/groups {"frequency": 3, "spread": 1}
6 uk,ac,rpms)/ {"frequency": 124, "spread": 1}
7 uk,co,bbc)/images {"frequency": 152, "spread": 3}
```

Fig. 1: Sample Profile in CDXJ Format

```
URI: https://www.news.BBC.co.uk/images/Logo.png?width=200&height=80&rotate=90#top
Canonical URL: news.bbc.co.uk/images/Logo.png?height=80&rotate=90&width=200
SURT URL: uk,co,bbc,news)/images/Logo.png?height=80&rotate=90&width=200
Registered Domain: uk,co,bbc)/
Segment Counts: {subdomain:1, path: 2, query: 3}
Path Initial: i
URI-Keys:
          H1P0: uk)/
          H3P0: uk,co,bbc)/
          HxP1: uk,co,bbc,news)/images
          DDom: uk,co,bbc)/
          DPth: uk,co,bbc)/1/2
          DIni: uk,co,bbc)/1/2/3/i
```

Fig. 2: Illustration of URI-Key

of the path. A generic template for this category of *URI-Keys* can be given as "`registered_domain)/[#subdomain[/#path[/#query[/path_initial]]]]`". The *DDom* policy includes only the registered domain name in Sort-friendly URI Reordering Transform (SURT) [14] format, while *DSub*, *DPth*, *DQry*, and *DIni* policies also include sections of the template up to `#subdomain`, `#path`, `#query`, and `path_initial` respectively.

Figure 2 illustrates the process of generating *URI-Keys* from a URI. URIs are first canonicalized then go through SURT. For *HmPn* policies, query section and fragment identifier of the URI are removed (if present), then depending on the values of "m" and "n" any excess portions from the SURT URL are chopped off. The hostname segments are given precedence over the path segments in a way that no path segment is added until all the hostname segments are included, hence `uk,co)/images` is an invalid *URI-Key*, but `uk,co,bbc,news)/images` would be valid if the hostname is `news.bbc.co.uk` or `www.news.bbc.co.uk`. For *DLim* policies, the registered domain name is extracted with the help of the Public Suffix list (which is updated periodically). Then depending on the individual policies, segments from zero or more sections (such as sub-domain and path) of the URI are counted, and if necessary, the initial letter of the first path segment is extracted (replaced with a "-" if not alphanumeric). These values are then placed inside the above template to form the key.

4 Implementation

We have implemented a *URI-Key* Generator, more than one CDX Profiler, and a script to merge profiles and published the code on GitHub[3]. We have also made the code available for analyzing and benchmarking the profiles. We plan to collect all the profiles generated from various places in a public repository, but our script currently generates local files and publishes them in the form of a public Gist[4] if configured to do so.

Our initial implementation used JSON [5] and JSON-LD [15] for profile serialization. It was good for small profiles, but for large profiles any data format that has a single root node (such as JSON, XML, or YAML) introduces many scale challenges. To make frequent lookup in these single root node profiles efficient, they need to be completely loaded into memory, which becomes a bottleneck. A

[3] `https://github.com/oduwsdl/archive_profiler`

[4] `https://gist.github.com/`

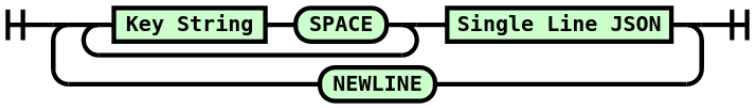


Fig. 3: Railroad Diagram: CDXJ File Format

single malformed character can make these profiles unusable. Hence a more linear key-value(s) based data formats such as CDX or ARFF [18] is more suitable in such cases as they allow an arbitrary split of data and enable easier profile merging. We have come up with a similar sort and index friendly file format "CDXJ" that is a fusion of CDX and JSON formats as illustrated in Figure 3 and utilized in Figure 1. The new CDXJ format also reduces the number of keys in a profile as it allows partial key lookup as opposed to the JSON format where we had to store all the intermediate smaller keys for higher level statistics.

Our initial implementation built the complete data structure in memory before serializing it to a file. We encountered a limitation in Python's dictionary implementation that degrades performance significantly when the number of keys in the dictionary is large. Hence, to make it scalable, we experimented with different profile generation optimization techniques such as preprocessing CDX files with standard Unix utilities (like grep, sort, uniq, sed, and awk) or using key-value databases (file based or in memory) for intermediate processing. The latter approaches involve more steps and setup, but scale well.

5 Evaluation

We generated 23 different profiles (17 *HmPn* policies, five *DLim* policies, and one *URIR* policy) for each archive dataset to measure their resource requirement and routing efficiency. To perform the analysis we prepared two types of datasets, archive profiles and query URI-Rs. For profiles, we used two archives:

- *Archive-It Collections* – We acquired the complete holdings of Archive-It [9] before 2013 and indexed the collections (in CDX format) to create a replica of the service. The archive has 2,952 collections with more than 5.3 billion URI-Ms and about 1.9 billion unique URI-Rs. Our Archive-It replica has more than 1.9 million ARC/WARC files that take about 230 TB disk space in compressed format. We created *URI-Key* profiles with various policies for the entire archive from the CDX files.
- *UK Web Archive* – We acquired a publicly available CDX index dataset from UKWA [16]. The dataset has separate CDX files for each year. We created individual *URI-Key* profiles from each of the early 10 years of CDX files (from year 1996 to 2005) with different profiling policies. We also created a combined profile by incrementally accumulating data for each successive year to analyze the growth. These 10 years of CDX files have about 1.7 billion URI-Ms and about 0.7 billion unique URI-Rs.

A second dataset was created by collecting three million URIs; one million random unique URI-R samples from each of these three sources:

- *DMOZ Archive* – URIs used in a study of HTTP methods [1].
- *IA Wayback Access Log* – URIs extracted from the access log used in a study of links to the Internet Archive (IA) content [2].
- *Memento Aggregator Access Log* – URIs extracted from the access log used in a previous archive profiling study [4].

5.1 Profile Growth Analysis

In commonly used CDX files each entry corresponds to a URI-M[5]. The length of each line in a CDX file depends on the length of the URI-R in it. Our experiment shows that the average number of bytes per line (α) in our dataset is about 275, which means every one gigabyte of CDX file holds about 3.9 million URI-Ms. Figure 4(a) can be used to estimate α in a CDX file. Equation 1 can be used to quickly estimate number of URI-Ms (C_m) in a large collection if total size of CDX files in bytes (S_c) is known.

$$C_m = \frac{S_c}{\alpha} \tag{1}$$

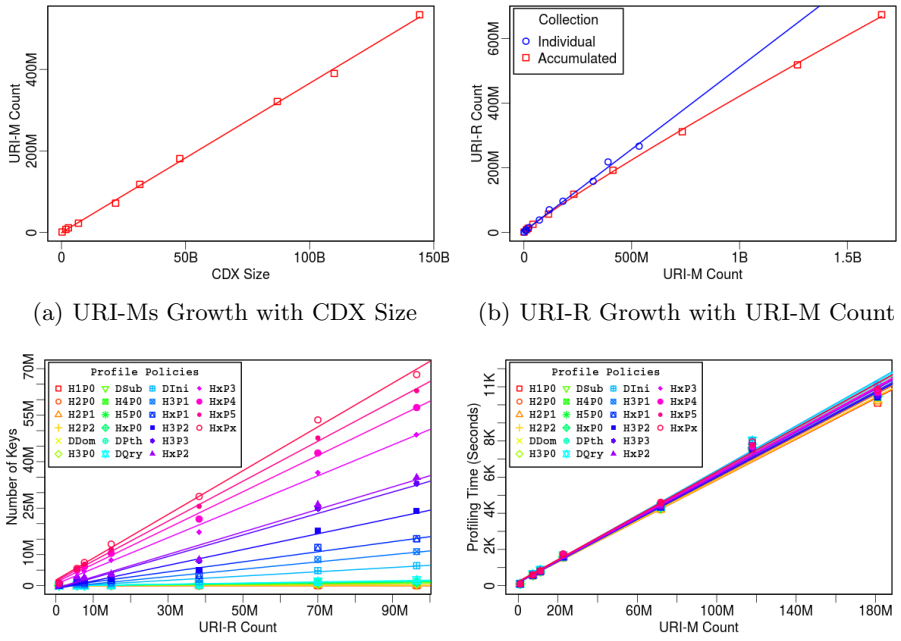
Figure 4(b) shows the relationship between URI-M Count (C_m) and URI-R Count (C_r) in two ways; 1) for each year of UKWA CDX files individually as if they were separate collections and 2) accumulated ten consecutive years of data one year at a time while each time it recalculates total number of unique URI-Rs visited. The ratio of URI-M Count to URI-R Count (γ) as shown in Equation 2 is indicative of the average number of revisits per URI-R for any given time period. The value of γ varies from one archive to the other because some archives perform shallow archiving while others revisit old URI-Rs regularly. In our dataset the value of γ is 2.46 for UKWA and 2.87 for Archive-It. The accumulated trend better accounts for how archives actually grow over time. It follows Heaps' Law [6] as shown in Equation 3 where URI-Rs are analogous to unique words in a corpus. K and β are free parameters that are affected by the value of γ, but the actual values are determined empirically. For the UKWA dataset K=2.686 and β=0.911.

$$\gamma = \frac{C_m}{C_r} \tag{2}$$

$$C_r = KC_m^{\beta} \tag{3}$$

URI-Keys are used as lookup keys in the *URI-Key* profile. The number of *URI-Keys* is a function of URI-Rs, not URI-Ms, hence increasing URI-Ms without introducing new URI-Rs does not affect the number of *URI-Keys*, instead, it only changes the value. Figure 4(c) shows the number of unique *URI-Keys*

[5] In our dataset Archive-It has 0.71% non-HTTP entries in their CDX files while UKWA has no non-HTTP entries.



(a) URI-Ms Growth with CDX Size (b) URI-R Growth with URI-M Count

(c) Space Cost (d) Time Cost

Fig. 4: Growth Costs Analysis for Different Profiling Policies

(C_h) generated for different numbers of unique URI-Rs (C_r) on different profiling policies. Every profiling policy follows a straight line with a slope value (ϕ_{policy}) and zero *Y-axis* intersect because for zero URI-Rs there will be zero *URI-Keys*. For a given profile policy, we define the ratio of *URI-Key Count* to *URI-R Count* as *Relative Cost* (ϕ_{policy}) as shown in Equation 4. The *Relative Cost* (ϕ_{policy}) varies from one archive to the other based on their crawling policy. Archives that crawl only a few URIs from each domain will have relatively higher *Relative Cost* than those who crawl most of the URIs from each domain they visit. Table 1 lists ϕ_{policy} values for UKWA dataset.

$$\phi_{policy} = \frac{C_h}{C_r} \tag{4}$$

Figure 4(d) illustrates the time required to generate profiles with different policies and different data sizes. Previously we used memory based, but now we use a file based profiling, which scales better and allows for distributed processing. Our experiment shows that the profiling time is mostly independent of the policy used, but we found that *DLim* policies take slightly more time than *HmPn* policies because they require more effort to extract the registered domain based on the public suffix list. We found that about 95% of the profiling time is spent on generating keys from URIs and storing them in a temporary file while the remaining time is used for sorting and counting the keys and writing the final

Table 1: Relative Cost of Various Profiling Policies for UKWA.

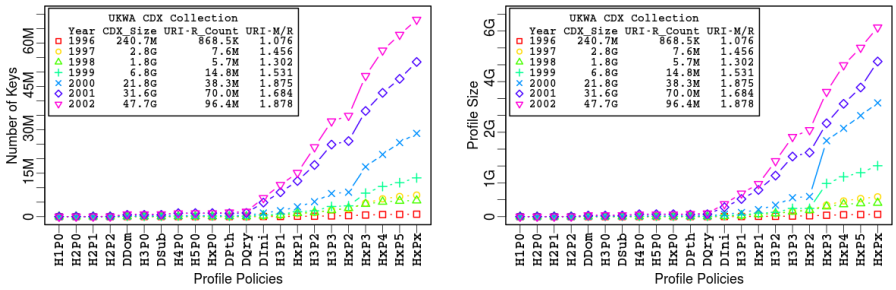
Policy	Rel. Cost	Policy	Rel. Cost	Policy	Rel. Cost
ϕ_{H1P0}	8.5e-07	ϕ_{H5P0}	0.01368	ϕ_{H3P3}	0.34668
ϕ_{H2P0}	0.00026	ϕ_{HxP0}	0.01371	ϕ_{HxP2}	0.36298
ϕ_{H2P1}	0.00038	ϕ_{DPth}	0.01577	ϕ_{HxP3}	0.49902
ϕ_{H2P2}	0.00056	ϕ_{DQry}	0.01838	ϕ_{HxP4}	0.58442
ϕ_{DDom}	0.00857	ϕ_{DIni}	0.06892	ϕ_{HxP5}	0.64365
ϕ_{H3P0}	0.00858	ϕ_{H3P1}	0.11812	ϕ_{HxPx}	0.70583
ϕ_{DSub}	0.00876	ϕ_{HxP1}	0.16247	ϕ_{URIR}	1.00000
ϕ_{H4P0}	0.01340	ϕ_{H3P2}	0.25379		

profile file. Hence a memory based key-value store can be used for the temporary data to speed up the process. Also, when an archive has a high value of γ, it might be a good idea to generate keys from each URI-R only once and multiply the keys with the number of occurrences of the corresponding URI-Rs, but when profiles are generated on small sub-sets of the archive there is a lower chance of revisits. Mean time to generate a key for one CDX entry τ can be estimated using Equation 5 where T is the time required to generate a profile from a CDX file with C_m URI-Ms. The value of τ depends on the processing power, memory, and I/O speed of the machine. On our test machine it was between 5.7e-5 to 6.2e-5 seconds per URI-M (wall clock time). As a result, we were able to generate a profile from a 45GB CDX file with 181 million URI-Ms and 96 million unique URI-Rs in it in three hours.

$$\tau = \frac{T}{C_m} \tag{5}$$

Figure 5 illustrates the correlation among the number of *URI-Keys* 5(a) and profile size on disk 5(b) for various collection sizes with various profiling policies. The policies are sorted in the increasing order of their resource requirement. If generated profiles are compressed (using gzip [7] with the default compression level), for bigger profiles they use about 15 times less storage than uncompressed profiles, but result in a similar growth trend. These figures are helpful in identifying the right profiling policy depending on the available resources such as storage, memory, computing power. Here are some common observations in these figures:

- For host segments less than three, path segments do not make a significant difference as they are not included unless all the host segments of the URI are already included.
- Keeping either the hostname or path segments constant while increasing the other shows growth in the value, but the growth rate decreases as the segment count increases.
- *DDom* profile shows quite the same results as *H3P0* profile.
- The last data-point (HxPx) shows a different trend, it is not just one path segment ahead of its predecessor (HxP5), but any path segments more than 5 are included in it.



(a) URI-Key Count. (b) Profile Size.

Fig. 5: Resource Requirement for Various Profiling Policies and Collection Sizes

- Growth due to path segments is significantly faster than the growth due to hostname segments.
- If a single path segment is to be included, it is better to include all host segments as it will not cause any significant resource overhead, but will provide better details.

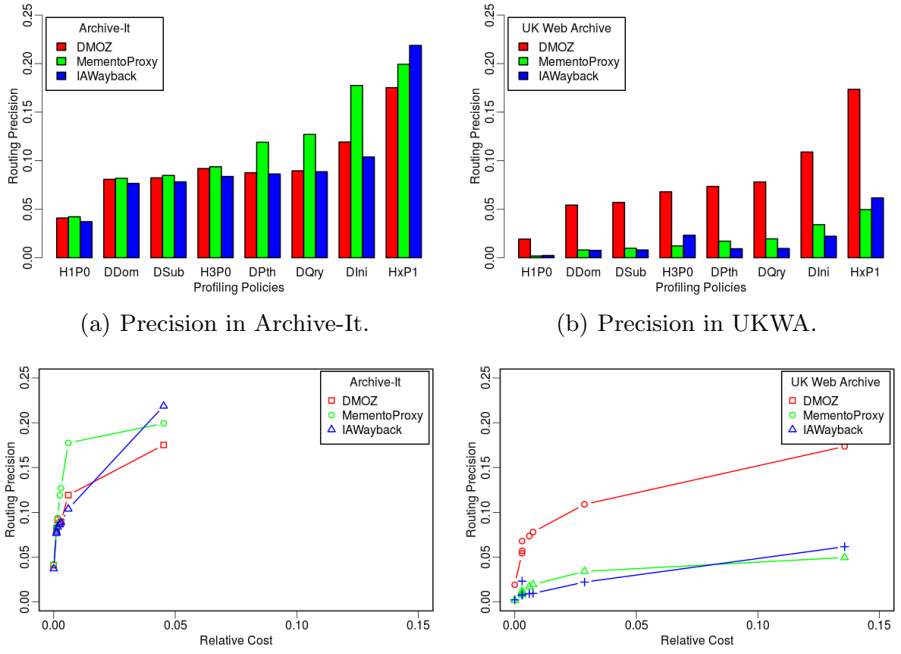
5.2 Routing Efficiency

To analyze the routing efficiency of profiles, we picked eight policies from the 23 policies we have used to generate profiles for both the archives in our dataset. These policies are *TLD-only* (*H1P0*), *H3P0*, *HxP1*, and all the five variations of the *DLim* policies. We then examined the presence of resources in the archives for our three query URI sample sets, each containing one million unique URI-Rs. Based on the profiling policy, a query URI-R is transformed into a *URI-Key* then it is looked up in the *URI-Key* profile keys to predict its presence.

To establish a baseline, we only check to see if the lookup key is present in the profile and do not use the statistical values (such as their frequency) present in the profile. This brings false positives in the result, but no false negatives, hence we do not miss any URI-Ms from the archive for the query URI-R. Table 2 is a good indicator of why archive profiles are useful: since both archives have $< 5\%$ of any of the query sets, there would be many unnecessary queries if an aggregator simply broadcasted all queries to all archives.

$$\text{Routing Precision}_{policy} = \frac{|\text{URI-R Present in Archive}|}{|\text{URI-R Predicted by Profile}_{policy} \text{ in Archive}|} \quad (6)$$

As illustrated in Figures 6(a) and 6(b), from both archives it can be seen that the *Routing Precision* (as defined in Equation 6) grows when a profile with higher *Relative Cost* is chosen. Figure 6 shows that *DDom* profile doubles the precision as compared to the *TLD-only* profile and the *HxP1* profile brings five fold increment in the routing precision. These figures also show that inclusion of sub-domain count does not affect the results much as there are not many



(a) Precision in Archive-It.

(b) Precision in UKWA.

(c) Precision and Cost in Archive-It.

(d) Precision and Cost in UKWA.

Fig. 6: Routing Precision of Different Profiling Policies in Different Archives.

domains that are utilizing more than one sub-domain. Figures 6(c) and 6(d) show that there is significant gain in *Routing Precision* with little increment in *Relative Cost*. The difference in the trends of Figures 6(c) and 6(d) can be understood by means of the following factors:

- Table 2 shows that less than 0.3% of the sample URIs from MementoProxy and IAWayback logs are archived in UKWA. This affects the growth in *Routing Precision* (on *Y-axis*) from one profile to the next.
- UKWA uses a shallow crawling policy which results in higher *Relative Cost*. This increases the distance (on *X-axis*) from one profile to the next.

A *URI-Key* profile is more robust than a *URIR* profile because it can predict the presence of resources in an archive that were added in the archive after the profile was generated. For example, if a new image `Large-Logo.png` is added under `bbc.co.uk/images`, it is very likely that the archives that crawl BBC will capture the new image. Without any update the *URI-Key* profile illustrated in Figure 1 will be able to predict the presence of the new resource, but a *URIR* profile will need an update to include the new URI-R before it can predict it.

Table 2: Presence of the Sample Query URI-Rs in Each Archive.

Archive	**DMOZ**	**MementoProxy**	**IAWayback**
Archive-It	4.097%	4.182%	3.716%
UK Web Archive	1.912%	0.179%	0.231%

6 Future Work and Conclusions

In this paper we have examined the space and precision trade-offs in different policies for producing profiles of web archives. We defined the term "URI-Key" to refer to the keys generated from a URI based on various policies that are used to track the distribution of holdings of an archive at different hostname and path depths or different segment counts. We found that the growth of the profile with respect to the growth of the archive follows Heaps Law, but the values of free parameters are archive dependent. We implemented a *URI-Key* generator and profiler scripts and published the code. We used CDX files from ODU's Archive-It replica and the UK Web Archive for generating profiles, and evaluated the profiles using a query set of three million URIs, created from one million URIs from each of DMOZ, IA Wayback access logs, and Memento Aggregator access logs. With precision defined as correctly predicting that the requested URI is present in the archive, we gained up to 22% routing precision without any false negatives with less than 5% relative cost as compared to the complete knowledge profile (*URIR* profile) that has both routing precision and relative cost 100%. The registered domain profile doubles the routing precision with respect to the *TLD-only* profile, while a profile with complete hostname and one path segment gives five fold routing precision. We found that less than 5% of the queried URIs are present in each of the individual archives. As a result, looking each URI up in every archive is wasteful. Hence, even a small improvement in routing precision can save a lot of resources and time in Memento aggregation.

Going forward, we plan to study the trade-off between the routing precision and recall by utilizing the statistical values stored against each key in the profile. We plan to develop a more sophisticated non-absolute statistical property to predict distribution of the holdings for various lookup keys that remains useful after merging multiple profiles. We plan to combine results of more than one type of profiles (such as *Time* and *URI-Key*) to improve the routing precision and recall. We also plan to create profiles of live archives from non-CDX sources, such as URI and keyword sampling to generate *URI-Key*, Language, Time, and Media Type profiles. We would also like to generate classification based profiles such as news, social media, game, and art. Finally, we would like to implement a production ready profile based Memento query routing system to be used in aggregators.

7 Acknowledgements

This work is supported in part by the International Internet Preservation Consortium (IIPC). Andy Jackson (BL) helped us with the UKWA datasets. Kris Carpenter (IA) and Joseph E. Ruettgers (ODU) helped us with the Archive-It data sets. Ilya Kreymer contributed to the discussion about CDXJ profile serialization format.

References

1. Alam, S., Cartledge, C.L., Nelson, M.L.: Support for Various HTTP Methods on the Web. Tech. Rep. arXiv:1405.2330 (2014)
2. AlNoamany, Y., AlSum, A., Weigle, M.C., Nelson, M.L.: Who and what links to the Internet Archive. International Journal on Digital Libraries 14(3-4), 101–115 (2014)
3. AlSum, A., Weigle, M.C., Nelson, M.L., Van de Sompel, H.: 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. pp. 60–71 (2013)
4. AlSum, A., Weigle, M.C., Nelson, M.L., Van de Sompel, H.: Profiling Web Archive Coverage for Top-Level Domain and Content Language. International Journal on Digital Libraries 14(3-4), 149–166 (2014)
5. Crockford, D.: The application/json Media Type for JavaScript Object Notation (JSON). RFC 4627 (Jul 2006)
6. Egghe, L.: Untangling Herdan's law and Heaps' law: Mathematical and informetric arguments. Journal of the American Society for Information Science and Technology 58(5), 702–709 (2007)
7. Gailly, J., Adler, M.: GZIP File Format. `http://www.gzip.org/` (2013)
8. Internet Archive: CDX File Format. `http://archive.org/web/researcher/cdx_file_format.php` (2003)
9. Internet Archive: Archive-It - Web Archiving Services for Libraries and Archives. `https://www.archive-it.org/` (2006)
10. ISO 28500: WARC (Web ARChive) file format. `http://www.digitalpreservation.gov/formats/fdd/fdd000236.shtml` (2009)
11. Mozilla Foundation: Public Suffix List. `https://publicsuffix.org/` (2015)
12. Sanderson, R.: Global Web Archive Integration with Memento. In: Proceedings of the 12th ACM/IEEE-CS Joint Conference on Digital Libraries. pp. 379–380. ACM (2012)
13. Sanderson, R., Van de Sompel, H., Nelson, M.L.: IIPC Memento Aggregator Experiment. `http://www.netpreserve.org/sites/default/files/resources/Sanderson.pdf` (2012)
14. Sigurðsson, K., Stack, M., Ranitovic, I.: Heritrix User Manual: Sort-friendly URI Reordering Transform. `http://crawler.archive.org/articles/user_manual/glossary.html#surt` (2006)
15. Sporny, M., Kellogg, G., Lanthaler, M.: A JSON-based Serialization for Linked Data. W3C Recommendation (2014)
16. UK Web Archive: Crawled URL Index JISC UK Web Domain Dataset (1996-2013). `doi:10.5259/ukwa.ds.2/cdx/1` (2014)
17. Van de Sompel, H., Nelson, M.L., Sanderson, R.: HTTP Framework for Time-Based Access to Resource States – Memento. RFC 7089 (Dec 2013)
18. Weka: Attribute-Relation File Format (ARFF). `http://weka.wikispaces.com/ARFF` (2009)