Aggregator Reuse and Extension for Richer Web Archive Interaction

Mat Kelly(✉)

Drexel University, Philadelphia, PA 19104, USA
mkelly@drexel.edu
https://matkelly.com

Abstract. Memento aggregators enable users to query multiple web archives for captures of a URI in time through a single HTTP endpoint. While this one-to-many access point is useful for researchers and end-users, aggregators are in a position to provide additional functionality to end-users beyond black box style aggregation. This paper identifies the state-of-the-art of Memento aggregation, abstracts its processes, highlights shortcomings, and offers systematic enhancements.

1 Introduction

Web archives act as a historical record of the web. The Internet Archive (IA) possesses the largest number of web archive holdings. These holdings are accessible through a set of interfaces to the Wayback Machine. Beyond IA, other web archives exhibit focused collection efforts, often providing unique captures within IA's temporal and spatial (i.e., URL [7]) voids [17]. A common usage pattern in accessing IA's captures is to request the archive's web site at archive.org, submit a URL of interest by providing it in a text input field, then selecting a date and time from the set of available captures for that URL in the past. This pattern may differ between web archives' respective web interfaces. Memento [27] provides the standards-based interoperable means, dynamics, syntax, and semantics for representing identifiers for archival captures (mementos) from a set of web archives. Each archive that supports the Memento Framework provides an HTTP endpoint for retrieving mementos from their respective archival holdings. Users can send a request for all captures of a URL to a variety of supporting archives through a single endpoint by an accessible tool that performs the logic of querying and combining results from multiple sources—a Memento aggregator.

Memento aggregators typically have reference to a set of endpoints to web archives that implement the Memento Framework. An aggregator may express this through a URI "template" like Fig. 1 or as a URI with an implicit append operation of a URI-R [27]. Upon receiving a request from a client with a parameterized URL (e.g., the URI-R applied to the template URI), an aggregator relays the argument received in this request as parameters for subsequent requests to each archive. When the aggregator receives a sufficient response,[1] as dictated

[1] This criteria is implementation-specific and may be associated with a temporal threshold, memento count, etc.

Y.-H. Tseng et al. (Eds.): ICADL 2022, LNCS 13636, pp. 313–328, 2022.
https://doi.org/10.1007/978-3-031-21756-2_25

```
t0: {scheme & hostname}/{resource type}/{format}/{URI-R}
t1: https://myarchive.org/timemap/link/http://example.com
m0: {scheme & hostname}/{datetime}/{URI-R}
m1: http://archive.md/20210619183508/https://icadl.net/icadl2021/
m2: https://archive.ph/eoQRZ
```

Fig. 1. An aggregator must be configured to supply parameters to an HTTP endpoint (like t_1), often exhibited in the form of a "templated URI" (t_0) for a URI-T as shown here. The suffixed red portion represents a URI-R http://example.com as used in practice. This URI templating is replicated (m_0) with URI-Ms (e.g., m_1), though a web archive need not identify its captures in this non-opaque manner (m_2 and m_1 identify the same memento). (Color figure online)

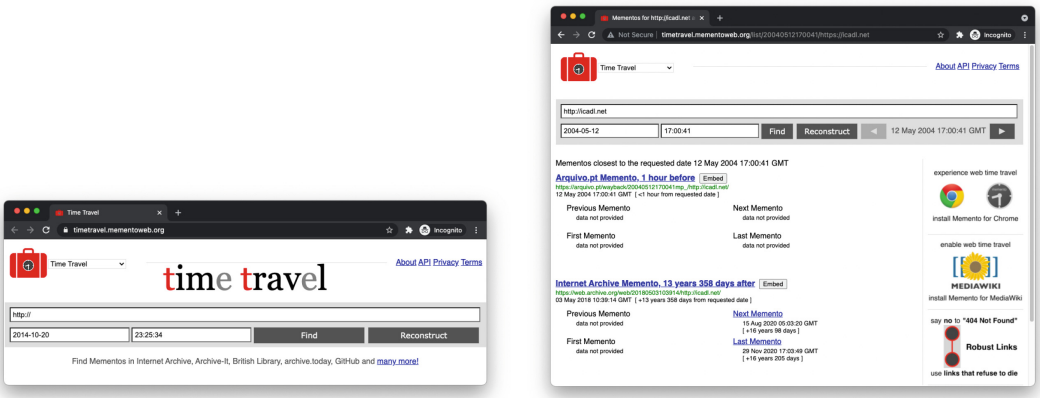
by the logic of the aggregator in-practice, the aggregator combines the results through a procedure that aligns with Memento syntax, often inclusive of temporal sorting.[2] The aggregator returns this "aggregated" response to the client. This description somewhat encompasses the conventional role of the aggregator. Its place as a means for users to interface with multiple web archives through a single request has the potential to be further utilized, exploited, and be more generally useful.

This paper examines the hierarchical (yet decoupled) relationship between a Memento aggregator and Memento-compliant web archives. While an aggregator and a set of archives often exhibit a static one-to-many relationship (respectively), there exists both more fundamental and more potentially complex hierarchies that may be exhibited using existing infrastructure. These exhibitions may be strategically and efficiently enhanced through consideration of this potential additional capability for the sake of enhancing the role of the aggregator in use cases for web archives. We build on existing work in defining a framework for aggregating public and private web archives [16]. Our focus will be on identifying (Sect. 6) and mitigating (Sect. 7) some outstanding issues both introduced by the framework as well as those that exist in current practice of interfacing with web archives using Memento aggregation.

2 Background

The Memento Framework [27] introduces the ability to perform temporal negotiation on the web by relating the current and past representations of a web page. Past representations are identified by "URI-Ms" and the original representation by a "URI-R", per Memento. Memento also introduces a resource to associate URI-Ms and URI-Rs through a structured listing called a TimeMap, identified by a "URI-T". A web archive may return a TimeMap representing its holdings, inclusive of URI-Ms, a URI-R, URI-Ts, and a URI-G for a "TimeGate". A TimeGate allows a client, through HTTP request headers, to specify a datetime basis for a likewise included URI-R. This paper relates to the information

[2] It is important to note here that TimeMaps do not need to be temporally sorted to be Memento compliant.



(a) (b)

Fig. 2. The "Time Travel" service provides a graphical, web-based endpoint to interface with LANL's Memento aggregator. After submitting a URI and date range in the interface (Fig. 2a), the results are displayed (Fig. 2b), showing the extent of the captures from a variety of pre-configured, server-defined web archives.

retrieval and relational aspects of Memento TimeMaps and not specifically to the temporal negotiation of Memento, the latter being a feature of TimeGates. We focus on the association of past and present URIs and not the ability to resolve the closest datetime, both of which Memento provides.

The concept of aggregation goes beyond the Memento specification by leveraging a similar structure to TimeMaps but allowing the URIs contained within the aggregated TimeMap to identify resources at multiple archives instead of a single archive. The Research Library at Los Alamos National Laboratory (LANL) deployed the original Memento aggregator [8,11], currently accessible through a web interface via the Time Travel service at https://timetravel.mementoweb.org/. This web service (Fig. 2a) provides an HTML form field for a user to specify the URI-R and a datetime then uses temporal negotiation to query a set of archives and return links to the results (Fig. 2b).

A central point of access also implies a central point of failure—if the aggregator goes down, no further aggregation may be performed, and users must again resort to querying individual web archives. In response, Alam and Nelson created MemGator [1], a portable, open-source, cross-platform, user-deployable Memento aggregator. This tool enables individuals to no longer solely rely on a single web-accessible aggregator but also configure, use, and potentially deploy their own. Also, unlike Time Travel, a user has the ability to control *which* web archives are queried for mementos. This newfound ability provided the accessibility of the aggregation capability to be further explored by researchers.

Memento is an extension to the Hypertext Transfer Protocol (HTTP). HTTP is a stateless, client-server based protocol on which the web is built. In the context of Memento, a client provides an HTTP request for a TimeMap of a URI in the past, often by appending a URI-R to a templated endpoint (Fig. 1). Both the

identifiers for a TimeMap and a memento are returned with corresponding Link [20] HTTP response headers giving additional context to the representation. A user (e.g., person) will typically act as a client through a user-agent (e.g., web browser, cURL[3]) and may send an HTTP request to a Memento aggregator with the expectation of receiving an HTTP response. The aggregator, in-turn, acts as a client to the web archives, relaying the request for the URI-R in the past and expects HTTP responses. This use case of a Memento aggregator playing the role of a server and a client is abridged in Sect. 7.4.

3 Related Work

Most research involving Memento aggregation relates to usage of the aggregator rather than enhancement of the aggregation process. In the same way that prior to MemGator, researchers would state "we requested URIs from the Time Travel Service", this statement was transformed to "we used MemGator to request URIs", indicative that it was useful for researchers to utilize their own aggregator instance [4,14,21]. A facet of this use case is the ability for researchers to customize the set of web archives to be used as the basis for querying, which is performed prior to running MemGator by modifying a configuration file.[4] This paper examines the aggregation process beyond accessing an aggregator and does so at a more abstract level than the ability to customize the archival sources.

3.1 Using Aggregators Beyond End-User Aggregation

As MemGator is free and open-source software (cf. Time Travel), many research endeavors on evolving the aggregation process have centered around enhancing its development beyond the limited endpoint-based Time Travel ecosystem. While the set of archives to be aggregated is static, both in accessing the Time Travel service as well as a deployed MemGator instance, other standards-based mechanisms like HTTP Prefer [26] provide a means of allowing a client to specify the set of archives aggregated to an "enhanced" aggregator—in this case, an extended version of MemGator [13]. This approach [13] entailed encoding the set of archives that normally reside in a server-side configuration file to be customizable at *query* time. The specification of custom archival sources utilizes the "Prefer" HTTP request header with a value being the self-describing, base-64 encoded JSON representing the aggregator's configuration of endpoints. A prototypical extension of MemGator referenced by the authors required the aggregator to read the HTTP request header and respond accordingly at runtime to request captures only from the archives specified by the client.

[3] https://curl.se/.

[4] An aside: researchers that need to control the process do so either through manipulation of their internal software (LANL experimenting with Time Travel [8]) or those outside of LANL utilizing MemGator.

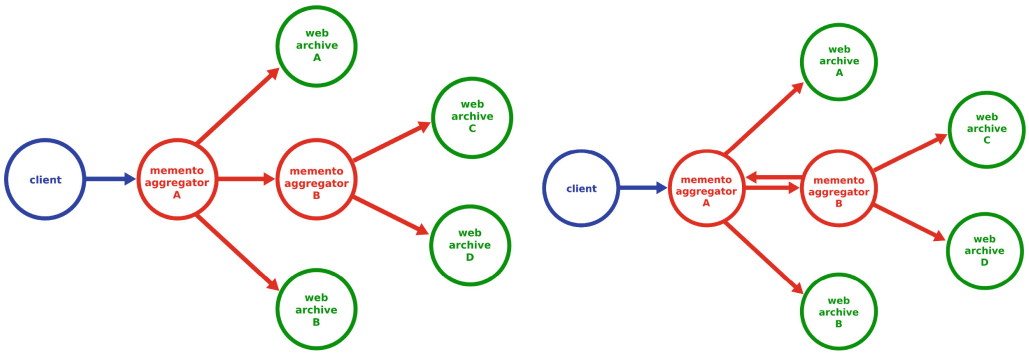


Fig. 3. An aggregator is configured to query HTTP endpoints (Fig. 3a), which are typically from web archives, but could equally be configured to be to other aggregators causing an "aggregator chaining" effect (Sect. 4.3). Aggregators are agnostic of whether their requester is a client, script, or aggregator itself (Fig. 3b) and thus may send a request that ultimately resolves to a requester causing an infinite loop.

3.2 Abstractions from Other Domains

The process of HTTP requests as recursively applied through an aggregator subsequently querying additional sources resembles a graph structure, typically reduced to a tree in the conventional case (Sect. 4.2). As this work reiterates the potential for an aggregator querying an aggregator [16], the scenario arises of graph-style cycles (Fig. 3) that must be mitigated. Additionally, we may encounter redundancies in this "chaining" process (Fig. 5) where aggregators down the request chain are configured to query identical, previously queried archives with the same parameters. The similarity of this problem resembles a singly linked list wherein a child does not know the capacity of its parent and is in adherence of HTTP being stateless. Here, an origin node is aware of that to which it links but a node is likely not aware of the linkages from its parent, to which the node itself is one.

3.3 Aggregation Optimization

The process of aggregation can be complex [19], both in programmatic logic to accomplish it as well as largely so in the temporal, spatial, and computational requirements. In conventional practice (Sect. 4.2), upon receiving a request, an aggregator will then send a request to each web archive, as defined by the endpoints in the aggregator's configuration. The process of sending these requests can typically be performed asynchronously [1], as the response time from a particular archive may be affected by a variety of factors including its infrastructure capabilities, the quantity of its holdings, the temporal spread of its holdings, etc.

Different web archives inherently possess a different set of archival holdings.[5] For example, an archive may only collect web pages within a limited set of ccTLDs [22] like `.ac.uk` and `.gov.uk` for academic and government websites in the United Kingdom (respectively). Repeated requests for TimeMaps from web archives that consistently have no mementos for a structured type of URI produce inefficiencies that are exacerbated when aggregated and affect the aggregation process. AlSum et al. [5] generated profiles to identify the distribution of URIs across archives and the effect on recall by both including and excluding IA from the aggregated results. MementoMap [3] provided an approach to remedy this issue with the cooperation of a web archive. By an archive supplying indexes of its holdings, a "map" can be created to abstractly represent (using wildcards) the extent of the holdings for specific URI patterns. This may be abstracted to the level of TLD (e.g., the extent of the holdings within the `.uk` TLD) down to the specificity of the quantity of holdings within a specific path of the URI. MementoMaps also provide a format to represent this extent both on the level of URI-R and URI-M. Through the cooperation of one such scoped archive, the Portuguese Web Archive, Alam et al. [3] were able to demonstrate the increase in efficiency of selectively sending requests to a subset of archives informed by their respective holdings. This work leveraged MemGator. Aturban et al. [6], through a longitudinal study on the web archives themselves, identified the disappearance of the base URI of an archive, further highlighting the need for an aggregator to be updated to ensure resolution as archives change their hostnames.

In related work, Bornand et al. [8] consulted logs from the aggregator created by the Time Travel service (the authors are from LANL) to create classifiers to effectively route queries rather than relying on a web archive to provide a profile. They analyzed over 1.2 million URI-Rs from the aggregator's cache (with over 239,000 URI-Ms) to identify a point-of-compromise for optimizing the requests sent to an archive based on the true and false positive rate as informed by prior requests.

Part of this work entails enabling the user to have more extensive interaction with web archives using Memento. This is frequently enabled through the use of browser extensions [15,25] and dedicated applications [12,18,28]. Mink[6] is an extension for the Chrome web browser that allows a user to extend the context of the web page they are currently viewing to be used as the basis of a request to a Memento aggregator. Some preliminary efforts have been performed to provide further user control over archival selection from the web browser using the extension, but have not been formalized nor deployed in the primary extension. Doing so entails either the approach of requiring an enhanced aggregator that receives a request to adapt their set of archives queried at runtime based on the user's request (a server-side approach) or for Mink to filter the results on the client after the aggregator returns the results. In the latter, client-side approach,

[5] We distinguish "archival holdings" from mementos in that the latter implies compliance with the Memento Framework.

[6] https://github.com/machawk1/mink.

the logic of aggregation becomes the responsibility of the extension when an aggregator does not comply with sending requests to archives outside of its base configuration.

4 Base Querying Models

Per Sect. 3, Memento aggregators are often configured to be used as a web service; in the case of MemGator, specifying a list of archives, timeouts, etc.; and "used" by querying the aggregator's HTTP endpoints with the URI as a parameter. In this Section we define aggregator "querying models" for further discussion.

4.1 Proxy-Style Querying (S_0)

An aggregator may be configured to query a single web archive. This is typically not exhibited because of redundancy (i.e., the user would normally just send the request to the archive directly), but serves as a base case for the querying models for further discussion. Here, the "aggregator" acts as a simple relay or proxy between the client and the web archive. This might potentially be useful for specifying a configuration to the aggregator beyond what can be expressed with a request to URI,[7] e.g., timeouts for a response.

4.2 Conventional Querying (S_1)

Typical aggregator usage entails a client sending a request to an aggregator that then queries multiple web archives, aggregates the responses, and returns this response to the client (Fig. 4). The internal logic of the aggregator is not necessarily as relevant in defining this model but is critical for an aggregator's operation. For example, an aggregator may pipeline the requests for more efficient querying. An aggregator also might require archives to respond within a time threshold and "short-circuit" the response to disregard archives that do not respond in time. The abbreviated set of results could then be aggregated based on the subset archives that have responded up to that point in time. Some of these aspects are discussed further in Sect. 7.

4.3 Aggregator Chaining (S_2)

A Memento aggregator may successfully query any endpoint that is Memento compliant. The response from an aggregator is itself also typically Memento compliant. This begets the possibility that what is typically considered a "web archive" configured as an endpoint to query by an aggregator may be an aggregator itself, i.e., an aggregator querying an aggregator (Fig. 3a). One reason this is not typically exhibited is because the set of archives that are queried are (in

[7] Tools like cURL can also specify timeouts as command-line flags, but this moves the responsibility to the client.

```
$ curl https://memgator.example/timemap/link/https://icadl.net/

<https://icadl.net>; rel="original",
<https://memgator.example/timemap/link/https://icadl.net>; rel="self";
type="application/link-format",
<https://web.archive.org/web/20180503103914/http://icadl.net/>; rel="first memento";
datetime="Thu, 03 May 2018 10:39:14 GMT",
<https://web.archive.org/web/20200815050320/https://icadl.net/>; rel="memento";
datetime="Sat, 15 Aug 2020 05:03:20 GMT",
<https://web.archive.org/web/20200826164340/https://icadl.net/>; rel="memento";
datetime="Wed, 26 Aug 2020 16:43:40 GMT",
<https://web.archive.org/web/20201101023226/https://icadl.net/>; rel="memento";
datetime="Sun, 01 Nov 2020 02:32:26 GMT",
<http://web.archive.org/web/20220602205625/https://icadl.net/>; rel="last memento";
datetime="Thu, 02 Jun 2022 20:56:25 GMT",
<https://memgator.example/timemap/link/https://icadl.net>; rel="timemap";
type="application/link-format",
<https://memgator.example/timemap/json/https://icadl.net>; rel="timemap";
type="application/json",
<https://memgator.example/timemap/cdxj/https://icadl.net>; rel="timemap";
type="application/cdxj+ors",
<https://memgator.example/timegate/https://icadl.net>; rel="timegate"
```

Fig. 4. A typical use case for a Memento aggregator is for a user to specify a URL and receive a TimeMap representing a list of identifiers (URI-Ms) in the past—S_1. Shown here is a Link [20] formatted aggregated TimeMap from MemGator containing a URI-R (line 3 in orange), URI-Ts (lines 4, 16–21 in green), URI-Ms (lines 6–15 in purple) and a URI-G (line 22 in blue). (Color figure online)

practice) manually validated before being put in-place in the configuration. In the case of the Time Travel service, there is no indication that an aggregator is queried by the basis aggregator handling the initial response. For MemGator, however, the set of endpoints is user-configurable, and thus this valid scenario may arise and has implications. The merits of "aggregator chaining" were discussed in the seminal work introducing the concept [16], but did not go into detail or highlight some problems that may occur. We reiterate and address these in Sect. 6.

As above, an aggregator may plausibly query a second aggregator. More fundamentally, and problematically, an aggregator can specify itself in its own definition of sources to query. This can be mitigated by the aforementioned manual validation, but the more scalable and programmatic approach might be accomplished through short-circuiting conditional logic in the querying function, i.e., preventing an aggregation web service from sending a request to itself and causing an infinite loop (Fig. 3b). Doing so in the self-referencing case is straight-forward but through the indirection introduced through aggregator, an "aggregator-in-the-middle" prevents this logic from being enforced, as a request from a secondary aggregator would be handled as if from any other client. We discuss this problem further in Sect. 7.2.

5 Core Features

In this paper we define approaches to extend the capability of the aggregator abstraction without regard to implementation. This brief but important Section defines the empirical assumptions and expectations currently exhibited by an aggregator. These premises of an aggregator set forth the foundational base cases of expectations of an implementation. We build on these assumptions in Sect. 7.

Expectation 1. An aggregator must treat web requests received as clients and the requests it sends to archival sources as agnostic of the dynamics of the receiver.

Expectation 2. An aggregator must treat clients' requests equally, regardless of whether a requestor is a user-agent, a script, or an aggregator itself.

Expectation 3. An aggregator is unaware of whether its own configuration incurs any sources queries of its parent.

Expectation 4. An aggregator must treat clients as stateless and return results from its queries sources.

6 Existing Problematic Scenarios

What might be deemed as "mis-"configuration of a Memento aggregator may only be exhibited and discoverable upon execution of a request for aggregation. Typical approaches for including a web archive as an aggregation source are (1) the popularity of the archive itself to merit inclusion, (2) manual discovery by those responsible for configuring the aggregator, or (3) efforts toward publicity on the part of the archive itself to make those responsible for the archive's existence and Memento compliance. There is no established process for an archive to declare the availability of its holdings in an effort to be included in a publicly accessible aggregator [23,24]. Web archives with restricted holdings may be unsuitable to aggregate for reason of privacy of the holdings [16] or the requirement to limit accessibility beyond the conventional public scope. For example, the UK Web Archive requires a client to be physically on-site to access some of its holdings, otherwise returning an HTTP 451 (Unavailable For Legal Reason) [9] status code.

Aggregators like the Time Travel service also supply TimeGate functionality, allowing for temporal negotiation (per Sect. 2), which is outside of this paper's scope. As temporal negotiation requires an index for efficient selection (required for scale cf. query time indexing), an aggregator would need to retain the extent of the captures on a URI-R basis from their set of sources. As this is dynamic due to the availability of various archives' web services, the non-static nature of the set of mementos in an archive, etc., a heuristic-based approach or some form of caching [8] might suffice for "good enough" temporal negotiation. For optimal precision of the representation of sources' holdings, runtime querying of said sources' respective indexes produces a more representative result. Thus, the

abstraction of a TimeGate service being co-located with an aggregator would still succumb to the effects described in this Section. The remainder of this Section describes three effects that can plague current aggregation instances: aggregation cycles (Sect. 6.1), self-reference (Sect. 6.2), and source redundancy (Sect. 6.3).

6.1 When a Tree Becomes a Graph

As an extension of S_2 in Sect. 4.3, an aggregator (A) requesting captures from a second aggregator (B) may cause a cycle if the latter aggregator is configured to query aggregator A. This can be mitigated using a few approaches, one of which we describe in Sect. 7.2. Figure 3b illustrates an abstract scenario where this might occur with user-configurable Memento aggregators.

6.2 Self-reference

A simpler example of the abstraction where an aggregator, through the request chain, is requested to respond to a request that it initiated is exhibited in an aggregator's own endpoints being within its configuration. A web service might be naive of the URI to which it is accessible, blindly sending responses after consuming and processing the parameters in the requests received. Likewise, the solution described in Sect. 7.2 would prevent this from occurring.

6.3 Duplication of Sources

The combination of aggregators being user-configurable and the potential for aggregators to query aggregators may result in duplication of results. For example, in Fig. 5, aggregator A queries web archive A, web archive B, and aggregator B. Aggregator B queries web archive A, web archive C, and web archive D. It could be useful for the clients of aggregator A to obtain the results from aggregator B, for instance, aggregator B may be privy to access restrictive web archives C and D. However, the results returned from aggregator B from web archive A will likely be redundant of those requested from aggregator A. Thus, the results may need to be deduplicated. This characteristic may also exist outside of aggregation. For instance, aggregators currently configured to request mementos from archive.org and archive-it.org (both hosted by Internet Archive) will often receive URI-Ms from each archive with precisely the same 14-digit time stamp represented in the URI-M. While it is possible that two services have unique captures (based on the tools used), this requires dereferencing the URI-Ms, which is out of the scope of this paper that focuses on TimeMaps.

7 Newfound Capabilities

In this paper we emphasize the contribution of the untapped functional potential of a Memento aggregator beyond simple aggregation. Section 5 outlined the

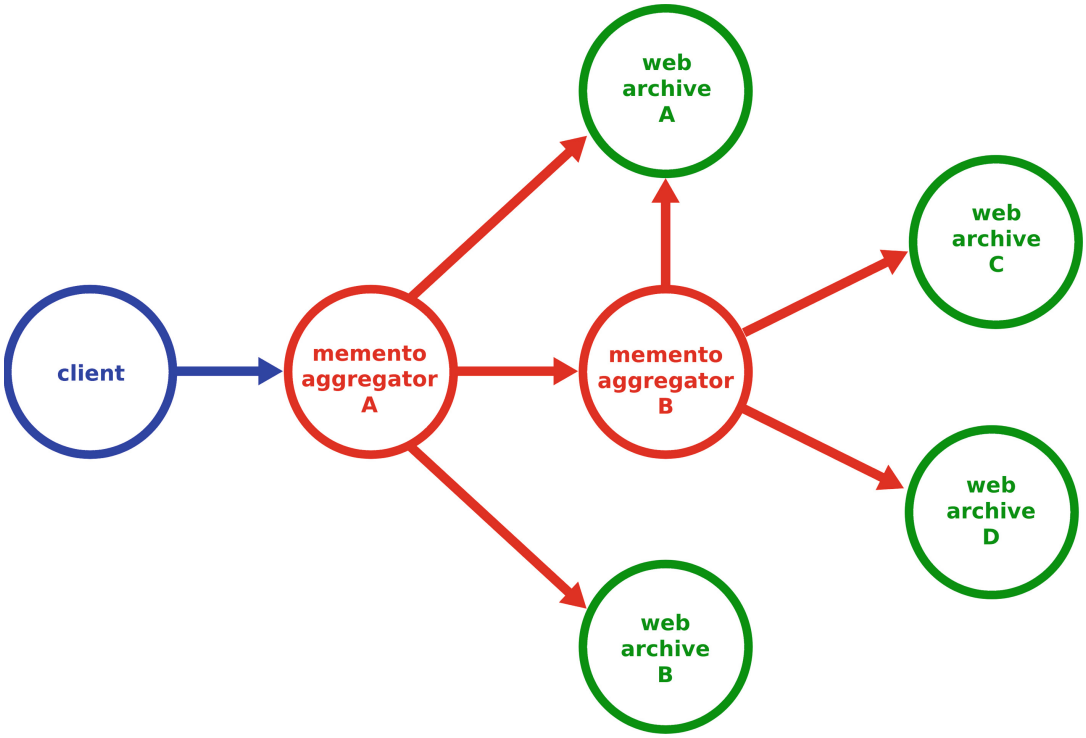


Fig. 5. An aggregator (A) configured to request captures from a set of sources $\{S\}$ inclusive of a second aggregator (B) can result with B redundantly querying one of A's sources, i.e., $|S_A \cap S_B| \geq 1$.

fundamental expectations of an aggregator that are exhibited and must be maintained as core functions. While the logic itself of strategically querying the set of archives with which an aggregator is configured has been explored in other works using profiles or machine-learning (Sect. 3.3), these do not consider the breadth of potential improvements like enabling the client to have further control of the aggregation beyond URI (e.g., using HTTP Prefer [13]), efficiency in returning partial results through HTTP endpoints, and mitigation of a non-curated set of archival sources.

7.1 User-Defined Set of Archives

HTTP provides a standardized means [13] for enabling the end-user (one querying an aggregator through HTTP) to specify the archival sources for aggregation – the HTTP Prefer request header [26]. The value for this header may include an encoded, modified version of the JSON data that is typically used to configure MemGator and contain custom values and transporting through the header. The expectation of an enhanced aggregator is that it will be required to decode this JSON and at its discretion, use that as the basis for the set of archives to query. Some nuances to this approach that have not been explored are (for example) whether the configuration can and should be applied to all users, the rules that should restrict which clients should be authorized to affect this change in the

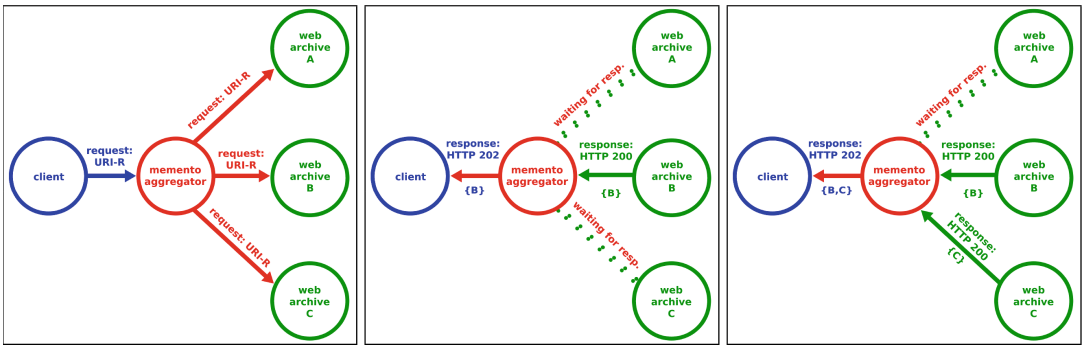


Fig. 6. Rather than an aggregator waiting for the slowest archival source to respond, the response can be progressively built based on the data received thus far. This response may be served to a client as a preliminary response as indicated by HTTP 202.

aggregator's operation, and how to further express the semantics to the extent to which the preference was applied (beyond supplying the Preference-Applied response header).

7.2 Cycle Detection

In Sect. 6.1, we introduced the potential for a cycle to occur when Memento aggregators are user-configurable and oblivious to the sources subsequently queried by aggregators further in the request chain. Approaches at mitigating cycles admittedly require the notion of HTTP being stateless to be violated. For instance, including a nonce or unique value to the request and propagating that to the sources queried (whether a web archive or aggregator), and likewise reading this value would allow the process to be short-circuited and provide a requestor some indication that the requestee was a requestor earlier in the hierarchical chain.

7.3 Preliminary Results Streaming

HTTP provides an often unused but standardized mechanism for a server to convey that a request is still processing (HTTP 202 status code) and that a client should wait and check back later [10], often at some indicated amount of time. In the context of Memento aggregation, web archives or other archival sources (e.g., other aggregators per Sect. 4.3), a set of sources from which resources are requested likely returns results in respectively varying amounts of time. This can create a bottleneck while the aggregation service waits for the slowest endpoint to respond but can be optimized by progressively building the result (Fig. 6). MemGator, for instance, merges TimeMaps as they arrive from the requesting aggregator and provide timeouts that can be specified by the user (i.e., the "user" that is executing the MemGator binary – not one making the HTTP request).

An important precondition for optimizing aggregators' processing through streaming is the recognition that Memento does not guarantee nor enforce internal temporal order of the identifiers in TimeMaps. When progressively merging TimeMaps from a partial set of sources requested, the merging process can be performed asynchronously relative to responses being received or more simply, not at all. For an aggregator to wait until all web archives have responded (which may never occur in the case of transient errors at an archive) is temporally inefficient. However, an incomplete (i.e., containing results only from a subset of archives), partially sorted, or unsorted aggregated TimeMap being returned to an end-user while an aggregator continues to wait can help to inform the end-user of the degree of success thus far. This may be potentially useful in cases where the results of the archives referenced in the aggregated TimeMap are explicit (e.g., through included metadata) instead of needing to be inferred (e.g., zero URI-Ms from an archive *might* mean no captures). This latter point can be helpful to end-users in making an informed decision to prematurely close the request if the results from an archive, as expressed in the partially aggregated TimeMap, are not to their expectations.

While the ability to return a TimeMap containing results from a subset of archives from which TimeMaps were requested may be useful and more efficient, the temporal burden for an aggregator to sort results is relatively less expensive, as it can be performed asynchronously and progressively. Despite this, partial, unsorted, concatenated TimeMaps returned using either a mechanism of streaming or through the HTTP 202 mechanism allows results, even if intermediate, to be immediately used rather than waiting on a likely unrevealed (to the end-user) set of conditions that are used prior to the response being returned.

7.4 Rescoping the Aggregator for Client-side Execution

In Sect. 2, we alluded to the propagation model, which may itself become recursive, of a client querying an aggregator that then similarly becomes the client through propagation of parameters. With Memento, a user-agent conventionally represents a client, transforming the request to the appropriate format (e.g., HTTP headers) as expected by a server (e.g., an aggregator).

From the client's perspective, the set of archives that an aggregator queried is not typically revealed. For example, if a client sends a request to an aggregator for `icadl.net` and receives back a TimeMap containing URI-Ms (Fig. 4), the set of archives represented by the URI-Ms *might* be representative of the entirety of the set, but that fact is not explicitly conveyed. It is likely and common, because of archival scoping and based on the URI-R provided, that archives within the set queried possessed no mementos for the URI-R and thus are not represented. It is wasteful and temporally inefficient to send requests to archives that possess no captures for a URI-R [16]. A priori knowledge as established by profiling archives of their holdings [2] or more specifically MementoMap [3], helps to mitigate this problem. These advancements allow the set of archives to be strategically defined so requests for URI-Rs that are unlikely to be in an archives' respective holdings are not requested. However, MementoMap requires archival cooperation and is

not foolproof if the index of the captures [8] is not updated to be representative of newly collected captures. It is also heuristic-based, so has false positive built in, i.e., likelihoods may result in no URI-Ms being returned in the TimeMap from an archive that was queried, despite their profile stating that they have captures.

8 Discussion and Future Work

Implicit to this work is the continuous effort to enable the end-user, for which aggregators are typically deployed, to be able to be more specific about that which they would like aggregated. As described in Sect. 3.3, allowing for this degree of interaction with a web service will likely have ramifications to efficiency, for example, caching mechanism may not be beneficial if archival sources vary with each request. For the Time Travel service, this might be moot, as the set of archives queried is controlled server-side. For open-source aggregators, however, which have the potential for extended capability, this process can be further optimized and explored.

There is also the notion of functional cohesion, that is, a service should ideally do one job and do it well. This cohesion is already violated in practice with the addition of TimeGate functionality being co-located with TimeMap querying (i.e., aggregation) endpoints. We hope to see further work done in investigating use cases for both the end-user querying aggregators, researchers deploying their own aggregators, and the functions and processes inherent to the aggregation procedure to enhance the capability to make the aggregation concept generally more usable.

9 Conclusion

This paper focused on the aspect of Memento aggregation. We identified the state-of-the-art in pure server-side aggregators (Time Travel) and user-deployable aggregators (MemGator). Through an aggregator being user-configurable and -deployable, which has proven useful to researchers, other potential issues may arise based solely on the current functionality of an aggregator. We proposed further functional extensions to the internal aggregation process.

From the perspective of a web service where a client sends an HTTP request to an endpoint, the aspects of this work may not much matter. However, the capacity of aggregators in the status quo still contains untapped potential capability beyond that the typical use case (S_1). By enumerating these potential concerns with a user-controlled Memento aggregator, the ultimate goal of enabling a client to have more expression and preference in the process of aggregating web archives will hopefully be improved.

Acknowledgement. For initial discussions on aggregator chaining and potential pitfalls, we would like to thank Chuck Cartledge, Sawood Alam, Michael Nelson, and Michele Weigle.

References

1. Alam, S., Nelson, M.L.: MemGator - a portable concurrent memento aggregator. In: Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL), pp. 243–244 (2016). https://doi.org/10.1145/2910896.2925452
2. Alam, S., Nelson, M.L., Van de Sompel, H., Balakireva, L.L., Shankar, H., Rosenthal, D.S.H.: Web archive profiling through CDX summarization. Int. J. Digit. Libr. **17**(3), 223–238 (2016). https://doi.org/10.1007/s00799-016-0184-4
3. Alam, S., Weigle, M.C., Nelson, M.L., Melo, F., Bicho, D., Gomes, D.: MementoMap framework for flexible and adaptive web archive profiling. In: Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL), pp. 172–181 (2019). https://doi.org/10.1109/JCDL.2019.00033
4. Alkwai, L., Nelson, M.L., Weigle, M.C.: Comparing the archival rate of Arabic, English, Danish, and Korean language web pages. ACM Trans. Inf. Syst. (TOIS) **36**(1), 1–34 (2017). https://doi.org/10.1145/3041656
5. AlSum, A., Weigle, M.C., Nelson, M.L., Van de Sompel, H.: Profiling web archive coverage for top-level domain and content language. Int. J. Digit. Libr. (3), 149–166 (2014). https://doi.org/10.1007/s00799-014-0118-y
6. Aturban, M., Nelson, M.L., Weigle, M.C.: Where did the web archive go? In: Proceedings of the Theory and Practice of Digital Libraries Conference (TPDL), pp. 73–84, September 2021. https://doi.org/10.1007/978-3-030-86324-1_9
7. Berners-Lee, T., Fielding, R.T., Masinter, L.: Uniform Resource Identifier (URI): generic syntax. IETF RFC 3986, January 2005
8. Bornand, N.J., Balakireva, L., Van de Sompel, H.: Routing memento requests using binary classifiers. In: Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL), pp. 63–72 (2016). https://doi.org/10.1145/2910896.2910899
9. Bray, T.: An HTTP status code to report legal obstacles. IETF RFC 7725, February 2016
10. Fielding, R.T., Reschke, J.F.: Hypertext Transfer Protocol (HTTP/1.1): semantics and content. IETF RFC 7231, June 2014
11. Jones, S.M., Klein, M., Van de Sompel, H., Nelson, M.L., Weigle, M.C.: Interoperability for accessing versions of web resources with the memento protocol. In: The Past Web, pp. 101–126. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-63291-5_9
12. Jordan, W., Kelly, M., Brunelle, J.F., Vobrak, L., Weigle, M.C., Nelson, M.L.: Mobile Mink: merging mobile and desktop archived webs. In: Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL), pp. 243–244 (2015). https://doi.org/10.1145/2756406.2756956
13. Kelly, M., Alam, S., Nelson, M.L., Weigle, M.C.: Client-assisted memento aggregation using the prefer header. Presented at the ACM/IEEE JCDL 2018 workshop on web archiving and digital libraries (WADL) (2018)
14. Kelly, M., Alkwai, L.M., Alam, S., Nelson, M.L., Weigle, M.C., Van de Sompel, H.: Impact of URI canonicalization on memento count. In: Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL), pp. 303–304 (2017). https://doi.org/10.1109/JCDL.2017.7991601
15. Kelly, M., Nelson, M.L., Weigle, M.C.: Mink: Integrating the live and archived web viewing experience using web browsers and memento. In: Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL), pp. 469–470 (2014). https://doi.org/10.1109/JCDL.2014.6970229

16. Kelly, M., Nelson, M.L., Weigle, M.C.: A framework for aggregating private and public web archives. In: Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL), pp. 273–282 (2018). https://doi.org/10.1145/3197026.3197045
17. Lobbé, Q.: Where the dead blogs are - a disaggregated exploration of web archives to reveal extinct online collectives. In: International Conference on Asian Digital Libraries (ICADL), pp. 112–123 (2018). https://doi.org/10.1007/978-3-030-04257-8_10
18. Nelson, M.L.: Right-click to the past - memento for chrome, October 2013. https://ws-dl.blogspot.com/2013/10/2013-10-14-right-click-to-past-memento.html. Accessed 1 Nov 2020
19. Nelson, M.L., Van de Sompel, H.: Adding the dimension of time to HTTP. In: Fagerberg, J., Mowery, D.C., Nelson, R.R. (eds.) The SAGE Handbook of Web History, chap. 14, pp. 189–214. SAGE Publications Ltd, 55 City Road (2019)
20. Nottingham, M.: Web linking. IETF RFC 8288, October 2017
21. Nwala, A.C., Weigle, M.C., Nelson, M.L., Ziegler, A.B., Aizman, A.: Local memory project: providing tools to build collections of stories for local events from local sources. In: Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL), pp. 219–228 (2017). https://doi.org/10.1109/JCDL.2017.7991576
22. Postel, J.: Domain name system structure and delegation. IETF RFC 1591, March 1994
23. Rosenthal, D.S.H.: The importance of discovery in memento, December 2010. https://blog.dshr.org/2010/12/importance-of-discovery-in-memento.html. Accessed 30 Nov 2020
24. Rosenthal, D.S.H.: Memento & the marketplace for archiving, January 2011. https://blog.dshr.org/2011/01/memento-marketplace-for-archiving.html. Accessed 30 Nov 2020
25. Sanderson, R., Shankar, H., Ainsworth, S., McCown, F., Adams, S.: Implementing time travel for the web. Code4Lib J. (13) (2011). https://journal.code4lib.org/articles/4979
26. Snell, J.M.: Prefer header for HTTP. IETF RFC 7240, June 2014
27. Van de Sompel, H., Nelson, M., Sanderson, R.: HTTP framework for time-based access to resource states - memento. IETF RFC 7089, December 2013
28. Tweedy, H., McCown, F., Nelson, M.L.: A memento web browser for iOS. In: Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL), pp. 371–372 (2013). https://doi.org/10.1145/2467696.2467764