Universal Distant Reading through Metadata Proxies with ArchiveSpark

Helge Holzmann
L3S Research Center
Appelstr. 9a
30167 Hannover, Germany
holzmann@L3S.de

Vinay Goel
Internet Archive
300 Funston Avenue
San Francisco, CA 94118, USA
vinay@archive.org

Emily Novak Gustainis
Center for the History of Medicine
Francis A. Countway Library of Medicine
10 Shattuck Street
Boston, MA 02115, USA
Emily_Gustainis@hms.harvard.edu

***Abstract*—Digitization and the large-scale preservation of digitized content have engendered new ways of accessing and analyzing collections concurrent with other data mining and extraction efforts. *Distant reading* refers to the analysis of entire collections instead of *close reading* individual items like a single physical book or electronic document. The steps performed in *distant reading* are often common across various types of data collections like books, journals, or web archives, sources that are very valuable and have often been neglected as *Big Data*.**

We have extended our tool ArchiveSpark, originally designed to efficiently process Web archives, in order to support arbitrary data collections being served from either local or remote data sources by using *metadata proxies*. The ability to share and reuse researcher workflows across disciplines with very different datasets makes ArchiveSpark a universal distant reading framework. In this paper, we describe ArchiveSpark's design extensions along an example of how it can be leveraged to analyze symptoms of *Polio* mentioned in journals from the *Medical Heritage Library*.

Our experiments demonstrate how users can reuse large portions of their job pipeline to accomplish a specific task across diverse data types and sources. Migrating an ArchiveSpark job to process a different dataset introduces an additional average code complexity of only 4.8%. Its expressiveness, scalability, extensibility, reusability, and efficiency has the potential to advance novel and rich methods of scholarly inquiry.

***Keywords*-Digital Libraries; Web Archives; Distant Reading**

I. Introduction

Books, journals, and other traditional print media items are being made available outside of physical libraries at a rapidly increasing pace. With the falling cost of high-quality digitization technologies, organizations such as *Google* and the *Open Content Alliance* are scanning books and other physical media on a massive scale and digitized content providers, such as *Internet Archive*, *The Digital Public Library of America*, *HathiTrust*, and *Europeana*, are enabling access to rare books, manuscripts, and other special collection materials otherwise available only at geographically sparse locations. Open access publishing and *Creative Commons* licensing are moving scholarly output in front of the paywall. In the case of **the Web**, organizations like the *Internet Archive* have developed automated crawlers that periodically capture Web content to ensure continued, public access to this dynamic and ephemeral content. The captured Web resources are timestamped to form persistent snapshots that are made accessible using tools like the *Wayback Machine*[1].

Given the **sheer volume and variety** of many of these collections, they may very well be considered *Big Data*, but are still widely neglected in this field. While digitized records can be read or viewed individually, their Big Data nature allows for completely new and interesting ways of access and analysis. Instead of *close reading* every single record, the digital collection can be filtered down by specific features, enriched, and aggregated for useful statistics in a **distant reading** manner [1].

Distant reading refers to the technique of analyzing large corpora of text documents without *close reading* every single one. Schulz [2] describes it as *"understanding literature not by studying particular texts, but by aggregating and analyzing massive amounts of data"*. The idea was proposed by Moretti [1] as a way to analyze texts systematically, using statistical and quantitative methods on texts, which are often expressed as networks of terms, where the edges represent the relationships among the terms. While it was originally meant to analyze literary fiction, we conceive it as a more general tool to derive information from big collections without reading single documents. Consider the example of *Polio* in a collection of medical articles. Using *distant reading* methods, we intend to analyze the most common symptoms of *Polio* and the body parts it affects. One way of doing this is to count how often the terms of interest occur in the documents mentioning *Polio*.

All of the above mentioned data sources have an important trait in common: they are maintained by libraries and archives, where records are **commonly organized in metadata indexes**. In Holzmann et al. [3], we presented ArchiveSpark[2], a scalable, expressive, and extensible framework, based on *Apache Spark*, that leverages such metadata records as lightweight data proxies to provide an

This work is partly funded by the European Research Council under ALEXANDRIA (ERC 339233)

[1] http://archive.org/web
[2] http://github.com/helgeho/ArchiveSpark

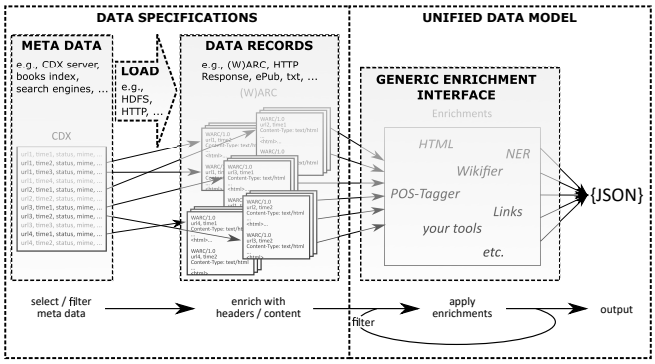

Figure 1: ArchiveSpark's selection and enrichment approach with variable metadata and data sources (cp. [3]). Dashed boxes illustrate the extensions and generalizations of the original approach for Web archives, towards a universal distant reading framework.

easy and efficient way to process *Web archives*. Web archives are typically large collections containing webpages and their embedded resources. To support lookups of a single records, Web archiving institutions generate and maintain indexes that contain pointers to the archived records. ArchiveSpark exploits these pre-existing index records, originally created for *close reading* scenarios, in order to support efficient filtering and processing on the complete dataset. Third-party tools can be easily integrated with ArchiveSpark to extract and/or derived new information to be used in *distant reading* workflows that are described in a rather declarative manner.

However, these workflows are not exclusive to the research of Web archives. They can, in fact, be applied to any digital collection. For example, the use of a natural language processing tool to annotate parts of speech in textual content is not limited to text from webpages or books but can be applied to any textual content regardless of its source. With that in mind, we redesigned and extended ArchiveSpark as illustrated in Figure 1, retaining and extending its two-step data access mechanism through *metadata proxies*, like indexes or search results. As shown in previous work, this approach is faster than alternative approaches in scenarios where only subsets of big data holdings are of interest, without depending on additional data stores [3].

In this paper, we describe the new concepts and extensions to ArchiveSpark that allows for various metadata sources and datasets, such as journals at the *Medical Heritage Library*[3] as well as Web archives loaded remotely from the Internet Archive's *Wayback Machine*[4].

II. Concepts and Architecture

ArchiveSpark provides efficient data access for use cases related to studying Web archives by leveraging metadata indexes [3]. In this section, we discuss the extension points of this approach and detail our new generic architecture to open it up to a wider variety of usage scenarios.

A. Beyond Web Archives

ArchiveSpark was designed as a framework for processing Web archives where the loaded collections consisted of metadata records stored in `CDX` (crawl/capture index) files, with the corresponding data records stored in `(W)ARC` (*Web archive files*). ArchiveSpark implemented the logic to load and parse `CDX` files as part of its core. When a user performed a task necessitating access to the original data records, it seamlessly read this data from the corresponding `(W)ARC` files. This random access pattern of reading data was supported by the inclusion of data record pointers (*filename* with record *offset* and record *length*) in the `CDX` metadata records.

In [3], we demonstrated the benefits of this two-step approach of first accessing metadata records as proxies of the corresponding data records and only accessing data records in a second step when required (cp. Figure 1). The approach was shown to be highly efficient for common data analysis tasks and there emerged a natural demand to implement this approach for other data types and sources. Given that, we laid out the following goals: **the ability to load Web archive data from different locations**, such as directly from the *Wayback Machine* without needing the records to be available on-site, and **the ability to support non-Web archive data**, like digitized books and journals. The only requirement was that all these collections would feature the same metadata/data record characteristics as the `CDX`/`(W)ARC` collections, where the metadata record included a pointer to the corresponding data record.

Remote access: Web archives are often enormous collections with sizes in the order of hundreds of terabytes and not every research institution has the capacity or the infrastructure to maintain their own local Web archive. However, there are institutions, such as the Internet Archive, that provide public access to their holdings. Thus, one of our extensions to ArchiveSpark was to support remote archives. This extension included the ability to load data not just from local disk but using different protocols, such as loading data over *HTTP* directly from the Wayback Machine. For the required metadata, queries can be made to the Internet Archive's *CDX Server*[5] that provides access to the metadata of every capture in the Web archive. An advantage of querying such a service to load metadata records is the ability to prefilter records based on criteria supported by the query service. We note that in the case of CDX Server, the retrieved metadata records do not include location information for the corresponding (W)ARC records. Access to the data records is through the Wayback Machine service

[3]http://www.medicalheritage.org
[4]http://web.archive.org
[5]https://github.com/internetarchive/wayback/tree/master/wayback-cdx-server

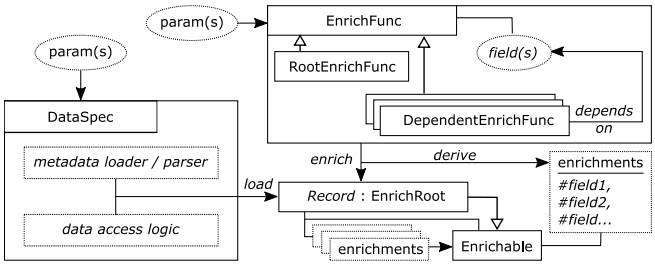

Figure 2: ArchiveSpark architecture.

by using the unique tuple key of *URL* and *timestamp*. With our extensions to ArchiveSpark, we can now support any possible combination of metadata and data record sources. By not requiring an ArchiveSpark user to possess a local copy of the data, we open up broader opportunities for data processing by multiple parties.

Interoperability: The aforementioned access patterns are not exclusive to Web archives but can be generalized to various other types of data as well. Examples include digitized books and journals, which may be stored in the form of files in a specific data format or made available through a public Web service and are therefore ready for being analyzed in a *distant-reading* manner with ArchiveSpark. We wanted ArchiveSpark to be able to make use of any query service, e.g., databases and search engines, that enables the retrieval of metadata with pointers to the corresponding data records. Furthermore, we sought interoperability to reuse existing components of ArchiveSpark like tools that have been bundled as modules to be used with Web archives. For instance, a sentiment analysis tool that has been prepared for use with ArchiveSpark for Web archives should be usable with any text, regardless of whether it is part of a Web archive or a book corpus or any other collection. By providing this flexibility, ArchiveSpark greatly facilitates sharing of code among researchers, even across different disciplines, as we show by our experiments in Sections IV.

B. Architecture Extensions

ArchiveSpark's new flexibility is achieved by the introduction of *data specifications* (`DataSpecs`). A `DataSpec` defines how metadata records are loaded and how they are associated to the corresponding data items. It encapsulates the code to load and access metadata as well as corresponding data records, while the details are abstracted away from the users. These isolated objects allow for easy sharing of datasets as well as data processing pipelines in the form of recipes, which are defined in a clean, declarative manner with the associated complex logic defined separately. At the same time, a given `DataSpec` is fully customizable by its developer and may be parameterized, which allows the user to easily specify required information, such as a data path or other types of location pointers. Figure 2 illustrates how `DataSpecs` fit into ArchiveSpark's architecture and play together with the other components.

Once ArchiveSpark is instructed to load a given `DataSpec`, the corresponding dataset is presented to the user as a collection of `EnrichRoot` records, the entry points into ArchiveSpark's data model. The initialization of such records is part of the used `DataSpec`. For instance, while a `DataSpec` for Web archives creates records that hold the corresponding `CDX` record information, one for book records will contain other kinds of metadata, like a book's title and author. Additionally, the specialized `EnrichRoot` records provide access to the actual data as defined by the used `DataSpec`.

Starting from there, ArchiveSpark's data model constitutes a *hierarchical tree structure* with `EnrichRoot` as the root node. Each node in this tree model is of type `Enrichable` and holds a value (for the root node, this is typically the metadata record), as well as zero or more child nodes with the child nodes representing either extracted or derived data based on the parent's value. Hence, the hierarchy of the data encodes its lineage, which corresponds to the dependency hierarchy of the applied functions.

Enrich functions (`EnrichFunc`) describe transformations that can be applied to the values stored in ArchiveSpark's data model. They encapsulate arbitrary logics and even third-party tools. Similar to a `DataSpec`, an `EnrichFunc` is used as an isolated blackbox in a declarative manner and can be configured by the user. All enrich functions, except for the root, have a dependency that determines their input, i.e., the dependency function's output will be the input of a `DependentEnrichFunc`. The very first `RootEnrichFunc` in such a dependency chain is in charge of loading the data of a record with accesses provided by the `EnrichRoot`.

To reuse and share an `EnrichFunc`, users can easily change its dependencies. As an example, consider a function that has been defined to depend on the body text of a webpage, i.e., its direct dependency is the `EnrichFunc` that extracts this text from the webpage. By changing this dependency dynamically to the function that extracts the title of a webpage, the same `EnrichFunc` will now operate on the title as input. Similarly, by changing the root dependency, a `EnrichFunc` can now by universally applied, regardless of the input data type, i.e., a function that has been developed to extract entities from text is now applicable to any text, regardless of whether it is text on a webpage or in a book.

III. Experiments and Evaluation

The aim of our experiments is to evaluate the applicability of the ArchiveSpark framework with the presented extensions to different data types and sources. In the following we outline an example to study the occurrence of Polio symptoms in a journal collection from the Medical Heritage Library (MHL). We then quantify commonalities of this job with similar tasks as well as different data types and sources, when defined with ArchiveSpark.

source	records accessed	# here
raw data	all	> 170,000
meta files	collection	3,148
full-text	matches	2,027

Table I: Comparison of accessed records with different metadata sources for the analysis of documents mentioning *Polio* in *State Medical Society Journals* in the *Medical Heritage Library*.

MHL is a digital curation collaborative effort among some of the world's leading medical libraries. It promotes free and open access to quality historical resources in medicine. The goal of the MHL is to provide the means by which readers and scholars across a multitude of disciplines can examine the interrelated nature of medicine and society, both to inform contemporary medicine and strengthen our understanding of the world. The MHL's growing collection of digitized medical rare books, pamphlets, journals, and films number in the tens of thousands, with representative works from each of the past six centuries, all of which are available through the Internet Archive and discoverable through an advanced full-text search interface[6].

A. Example Study: Polio

In this example we would like to demonstrate a possible distant-reading scenario, using ArchiveSpark's new concepts as introduced in Section II-B with a non-Web archive dataset. For that, we exemplarily analyze a subset of the entire MHL corpus with more than 170,000 documents, the *State Medical Society Journals* collection, to study symptoms and parts of the body mentioned together with *Polio*. The required data specifications for this tasks are provided in a public open-source project[7], which currently includes three `DataSpecs` for different combinations of metadata and data source:

- **`MhlHdfsSpec` (with local data)**: Metadata and data records loaded from local (distributed) files.
- **`MhlHdfsSpec` (with remote data)** Local metadata with contents on remote servers loaded on demand.
- **`MhlSearchSpec`** MHL's search system is queried for metadata, records are loaded from the Internet Archive.

Table I gives an overview of the records accessed when using the different metadata sources as defined by these specs. If we worked on the raw data, i.e., not using ArchiveSpark, we would need to scan all MHL documents. By incorporating local metadata files (`MhlHdfsSpec`) as metadata proxies to filter for the collection of interest, we can significantly reduce the number of accessed records. With the use of full-text search (`MhlSearchSpec`), we receive an even stronger gain in efficiency as only those records that match our criterion will be accessed by ArchiveSpark.

To begin our analysis, we load the dataset of interest, using ArchiveSpark, by passing in a data specification that defines a query to MHL's full-text search system (`rdd` denotes the work representation of the dataset, `sc` refers to the *Apache Spark Context* of this job, which is given):

```
  val query = MhlSearchOptions(
query = "polio",
collections = MhlCollections
.Statemedicalsocietyjournals)

  val rdd = ArchiveSpark
.load(sc, MhlSearchSpec(query))
```

After this point, all instructions are not specific to MHL anymore. By simply changing the loading part, **the same analysis can be run on any similar dataset**.

In the next steps, we define our set of symptom-related terms that we would like to count in the documents, followed by an in-line definition of the required *enrich function*. This `EnrichFunc`, called `symptoms`, depends on the lower case version of the text and enriches it with the subset of symptoms contained in the text. Only at this point does ArchiveSpark integrate the actual text for the records in the selected dataset, which is included in the metadata proxies.

```
  val symptomSet = Seq("extremity",
"neck", "vomiting", "fever", "headache",
"irritability", "abdominal", "lethargy")

  val symptoms = LowerCase
.map("symptoms") {text: String =>
symptomSet.filter(text.contains)}

  val enriched = rdd.enrich(symptoms)
```

By printing a record of this enriched dataset in *JSON* format (`enriched.peekJson`), we can see the lineage of the applied enrichments with symptoms contained in this document (abridged):

```
{
  "text": {
    "lowercase": {
      "symptoms": [
        "headache", "neck", ...
      ]
    }
  }
}
```

Finally, the identified symptoms are counted and aggre-

[6] http://mhl.countway.harvard.edu/search

[7] https://github.com/helgeho/MHLonArchiveSpark

gated among all documents, as visualized in Figure 3:

```
  val symptomCounts = enriched
.flatMapValues(symptoms).countByValue
```

IV. Interoperability

The primary goal of our extensions to ArchiveSpark was to make it an universal tool for working with digital library collections. This entailed providing compatibility with different types of data collections and supporting the sharing of workflows across disciplines. Users would be able to easily rerun their analysis or reuse tools that have been prepared for ArchiveSpark on their own computing infrastructure, specific to their research needs. ArchiveSpark achieves this by abstracting away, to a large degree, platform, tool and data specific parts of the workflow so that the code can primarily focus on implementing the user's task. Implementation details are encapsulated as part of core ArchiveSpark or in separate modules, i.e., `DataSpecs` and `EnrichFuncs` (s. Sec II-B).

1) Experimental Setup: **Measuring the complexity of code** is not a straight-forward process as it needs to guard against variations in coding style and unnecessarily complex code structures. Many metrics exist that either try to capture the length of the source code, e.g., *lines of code*, or its structure, e.g., *cohesion / coupling*, nicely compared and evaluated by Yu and Zhou [4]. However, none of them meets our needs of measuring the *reduction of code* when reusing data processing workflows with ArchiveSpark for different data sources or in similar jobs. We are rather interested in the number of function calls that are required to define a workflow (cp. Sec. III-A). This can be assessed from the perspective of ArchiveSpark in two levels: (1) what is abstracted away from the user and hence, taken care of by ArchiveSpark, i.e., *internal*, (2) what does the user need to write in order to specify a job, i.e., *external* function calls.

We are interested in function calls that originate in one of these layers with the target of invocation outside the layer. Such invocations were analyzed using Java's included profiling tool `HPROF`[8] with a stack depth of 10 (Java parameter `-agentlib:hprof=depth=10`).

The reusability of code among jobs was measured by treating all identified across-abstraction-layer calls of a job as sets of the target functions suffixed with numbers for each call. The particular data is irrelevant for this evaluation as we intend to focus only on job definitions, the same function calls are counted only once, irrespective of the parallelism of a job or how big the data is.

Evaluated Workflows. The following jobs have been analyzed in our evaluation:

1) **MHL Smoke:** minimalistic example with MHL's search and content access.

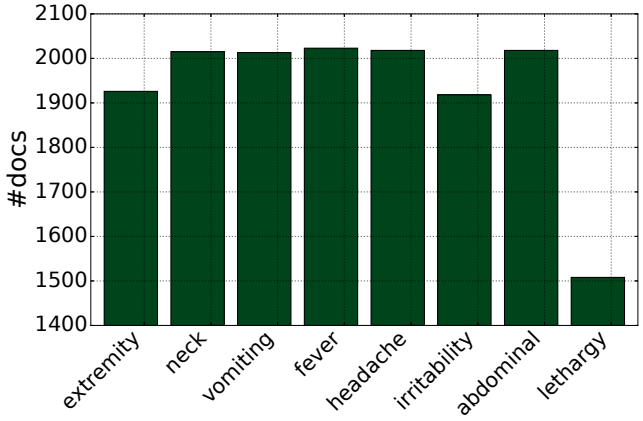

Figure 3: Distribution of symptoms and affected body parts in documents mentioning *Polio* in State Medical Society Journals at the Medical Heritage Library.

2) **MHL Polio:** example from Sec. III-A.
3) **MHL E. Search:** MHL's API to filter by collection, counting entities instead of symptoms[9].
4) **MHL E. Local:** metadata and content from local files, counting entities.
5) **MHL E. Remote:** local metadata and remote content from the Internet Archive, counting entities.
6) **Web E. Local:** Web archive filtered by domain and year with local files (CDX/WARC), counting entities.
7) **Web E. Wayb.** CDX server to filter by domains, content from Wayback Machine, counting entities.

2) Evaluation Results: Table II gives an overview of the extent of reusability among the job workflows in our evaluation. This is quantified by the number of additional function calls needed when transforming a small job, serving as a template, to a larger one. As an example, compare *Mhl E. Search* and *Web E. Wayb*, both counting mentioned entities in the corresponding collections. As the latter one is smaller, we consider it to be our starting point for transforming it to now using MHL documents as the data source instead of web captures from the Wayback Machine. This process requires 5 additional function calls in the job definition, while internally the complexity is increased by 224. That is only 2% in this case and overall 4.8% on average, which is a very small fraction of the abstracted internal counterparts.

Thus, with ArchiveSpark we can reuse large parts of data pipelines to accomplish a certain job, even across different data sources. This makes the case for sharing of analysis and distant reading workflows in the form of recipes as well as modules, such as `DataSpecs` and `EnrichFuncs`, to be reused among researchers and disciplines.

[8]http://docs.oracle.com/javase/7/docs/technotes/samples/hprof.html

[9]https://github.com/helgeho/FEL4ArchiveSpark

intern. / *extern.*	**MHL Polio**	**MHL E. Search**	**MHL E. Local**	**MHL E. Remote**	**Web E. Local**	**Web E. Wayb.**
MHL Smoke	67 / 14 (21%)	136 / 14 (10%)	195 / 16 (8%)	182 / 16 (9%)	225 / 17 (8%)	194 / 17 (9%)
MHL Polio		71 / 3 (4%)	217 / 7 (3%)	199 / 7 (4%)	195 / 7 (4%)	206 / 7 (3%)
MHL E. Search			247 / 4 (2%)	226 / 4 (2%)	214 / 5 (2%)	224 / 5 (2%)
MHL E. Local				24 / 0 (0%)	158 / 3 (2%)	129 / 3 (2%)
MHL E. Remote					148 / 3 (2%)	122 / 3 (2%)
Web E. Local						87 / 1 (1%)

Table II: Pairwise added complexity among jobs with respect to *external* user code and *internal* function calls.

V. Related Work

We presented a comprehensive overview of related works on Web archive access methods when we published ArchiveSpark in Holzmann et al. [3]. Back then, we argued that *specialized* approaches like search engines for Web archives are not suitable for building research corpora and detailed data analysis, as their filtering capabilities are not flexible enough and limited to pre-defined lookups. With the new design concepts introduced in Section II, ArchiveSpark can now make use of retrieval systems, such as *Tempas* (*Temporal Archive Search*) [5, 6, 7] to pre-filter records, as demoed in [7]. This ability of ArchiveSpark to plug in existing systems and its generalized support across different data types and sources makes it a universal tool for *distant reading* and to the best of our knowledge, it is the first one of its kind. A good overview of the related works in distant reading from the visual analytics perspective for *Digital Humanities* has been published by Jänicke et al. [8]. The integration of similar concepts with search was discussed by Jackson et al. [9], by providing visualizations on top of traditional Web search results list to support scholarly activities. Although very convenient, such an approach would require an additional index and is again limited by the pre-built capabilities of such a system.

Looking at the field of **Big Data**, ArchiveSpark may be of use for as well as benefit from various related works. Maemura et al. [10] presented a framework to document the research process in Web archives, contributing to a better understanding of the findings and their provenance, in order to reuse of data, methods, and workflows. ArchiveSpark provides a technical solution to this by enabling a rather declarative description of data processing workflows that can be reused among different datasets, even beyond Web archives. This supports new processes to work with big data across teams and projects in coordination as well as data sharing, as sought by Saltz [11]. At the same time, we tackle interoperability challenges, widely present when dealing with *Big Data* [12].

VI. Conclusion and Outlook

ArchiveSpark was originally developed as a tool for use by scholars to access and work with Web archives. With the extensions presented in this paper it has been extended to serve as a universal tool for data analysis and distant reading across different data types and sources. We note that all advantages and features of the first version of ArchiveSpark, e.g., gains in performance through the inclusion of metadata and the implicit lineage documentation of derived information, have carried over to this new version.

Our experiments show the extent of code reusability achieved by ArchiveSpark. On average, swapping in new data sources and tools on shared workflows results in an added complexity of only 4.8% compared to the internal instruction set of our framework. This advantage of reusability provided by ArchiveSpark can be exploited in the future to provide recipes for different analysis tasks that can be easily customized for individual needs.

References

[1] F. Moretti, *Distant Reading*. Verso Books, 2013.

[2] K. Schulz, "What is distant reading," *The New York Times*, vol. 24, 2011.

[3] H. Holzmann, V. Goel, and A. Anand, "ArchiveSpark: Efficient Web Archive Access, Extraction and Derivation," in *JCDL'16*.

[4] S. Yu and S. Zhou, "A survey on metric of software complexity," in *ICIME'2010*.

[5] H. Holzmann and A. Anand, "Tempas: Temporal Archive Search Based on Tags," in *WWW'16*.

[6] H. Holzmann, W. Nejdl, and A. Anand, "On the Applicability of Delicious for Temporal Search on Web Archives," in *SIGIR'16*.

[7] ——, "Exploring web archives through temporal anchor texts," in *WebSci'17*.

[8] S. Jänicke, G. Franzini, M. F. Cheema, and G. Scheuermann, "On close and distant reading in digital humanities: A survey and future challenges," *Proc. of EuroVisSTARs*, pp. 83–103, 2015.

[9] A. Jackson, J. Lin, I. Milligan, and N. Ruest, "Desiderata for exploratory search interfaces to web archives in support of scholarly activities," in *JCDL'16*.

[10] E. Maemura, C. Becker, and I. Milligan, "Understanding computational web archives research methods using research objects," in *BigData'16*.

[11] J. S. Saltz, "The need for new processes, methodologies and tools to support big data teams and improve big data project effectiveness," in *BigData'15*.

[12] A. Kadadi, R. Agrawal, C. Nyamful, and R. Atiq, "Challenges of data integration and interoperability in big data," in *BigData'14*.