Building and Querying Semantic Layers for Web Archives

Pavlos Fafalios, Helge Holzmann, Vaibhav Kasturia, Wolfgang Nejdl
L3S Research Center, Hannover, Germany
{fafalios, holzmann, kasturia, nejdl}@l3s.de

***Abstract*—Web archiving is the process of collecting portions of the Web to ensure that the information is preserved for future exploitation. However, despite the increasing number of web archives worldwide, the absence of efficient and meaningful exploration methods still remains a major hurdle in the way of turning them into a usable and useful information source. In this paper, we focus on this problem and propose an RDF/S model and a distributed framework for building semantic profiles ("layers") that describe semantic information about the contents of web archives. A semantic layer allows describing metadata information about the archived documents, annotating them with useful semantic information (like entities, concepts and events), and publishing all this data on the Web as Linked Data. Such structured repositories offer advanced query and integration capabilities and make web archives directly exploitable by other systems and tools. To demonstrate their query capabilities, we build and query semantic layers for three different types of web archives. An experimental evaluation showed that a semantic layer can answer information needs that existing keyword-based systems are not able to sufficiently satisfy.**

I. Introduction

Significant parts of our cultural heritage are produced and consumed on the Web. However, the ephemeral nature of the Web makes most of its information unavailable and lost after a short period of time. Aiming to avoid losing important historical information, a web archive captures portions of the Web to ensure the information is preserved for future researchers, historians, and interested parties in general.

Despite the increasing number of web archives worldwide, the absence of efficient and meaningful exploration methods still remains a major hurdle in the way of turning web archives into a usable and useful source of information. The main functionalities offered by existing systems are to find older versions of a specific web page, to search on specific collections, or to search using keywords and filter the retrieved results by selecting some basic metadata values. However, for a bit more complex information needs, which is usually the case when exploring web archives, keyword-based search leads to ineffective interactions and poor results [1]. This is true especially for *exploratory search* needs where searchers are often unfamiliar with the domain of their goals, unsure about the ways to achieve their goals, or need to learn about the topic in order to understand how to achieve their goals [2]. Thus, for exploring web archives, there is the need to go beyond keyword-based search and support more advanced information seeking strategies [1], [3].

To cope with this problem, we propose building semantic profiles ("layers") that describe semantic information about the contents of archived documents. Specifically, we base upon Semantic Web technologies and propose an RDF/S [4] data model that allows: a) describing useful metadata information about each archived document, b) annotating each document with entities, concepts and events extracted from its textual contents, c) enriching the extracted entities[1] with more semantic information (like properties and related entities coming from other knowledge bases), and d) publishing all this data on the Web in the standard RDF format, thereby making all this information directly accessible and exploitable by other systems and tools. Then, we can use that model for creating and maintaining a semantic repository of structured data about a web archive. Note that the actual contents of the web archive are not stored in the repository. The proposed approach only stores metadata information that allows identifying interesting documents and information based on several aspects (time, entity, type or property of entities, etc.). Therefore, such a repository acts as a *semantic layer* over the archived documents.

By exploiting the expressive power of SPARQL [5] and its federated features [6], we can run advanced queries over a semantic layer. For example, in case we have constructed a semantic layer for a *news archive*, we can run queries like:

- find articles of 1995 discussing about New York lawyers
- find medicine-related articles published during 1995
- find out the most discussed politician during 1995
- find out politicians discussed in articles of 1990 together with *Nelson Mandela*

Note that for all these queries we can directly (at query-execution time) integrate information coming from online knowledge bases like DBpedia [7]. For instance, regarding the first query, for each lawyer we can directly access DBpedia and retrieve his/her birth date, a photo and a description in a specific language. Thus, semantic layers enable connecting web archives with existing knowledge bases.

In a nutshell, in this paper we make the following contributions:

- We introduce a simple but flexible RDF/S data model, called *Open Web Archive*, which allows describing and publishing metadata and semantic information about the contents of a web archive.
- We detail the process of constructing semantic layers and we present an open source and distributed framework, called *ArchiveSpark2Triples*, that facilitates their efficient construction.
- We present (and make publicly available) three semantic layers for three different types of web archives: one for

[1] For simplicity, when we say *entity* we refer to *entity*, *concept* (e.g., Democracy) or *event*.

a *versioned web archive*, one for a *non-versioned news archive*, and one for a *social media archive*.

- We present the results of a comparative evaluation using a set of 20 information needs of exploratory nature (providing also their relevance judgements). The results showed that a semantic layer can satisfy information needs that existing keyword-based systems are not able to sufficiently satisfy. They also enabled us to identify problems that can affect the effectiveness of query answering.

The rest of this paper is organized as follows: Section II motivates our work and presents related literature. Section III introduces the *Open Web Archive* data model and describes the process and a framework for constructing semantic layers. Section IV presents three semantic layers for three different types of web archives, as well as their query capabilities. Section V presents evaluation results. Finally, Section VI concludes the paper and discusses directions for future research.

II. Motivation and Related Work

In this section, we first motivate our work by discussing information needs that our approach intends to satisfy for enabling more sophisticated search and exploration of web archives. Then we review related works by discussing also the difference of our approach.

A. Motivating questions

Working with large web archives in the context of the Alexandria project[2], we have identified the following information needs that an advanced exploration system for web archives should satisfy:

Q1 *Information Exploration.* How to explore documents about entities from the past in a more advanced and "exploratory" way, e.g., even if we do not know the entity names related to our information need? For example, how can we find articles of a specific time period discussing about a specific category of entities (e.g., *philanthropists*) or about entities sharing some characteristics (e.g., *born in Germany before 1960*)?

Q2 *Information Integration.* How to explore web archives by also integrating information from existing knowledge bases? For example, how can we find articles discussing about some entities and for each entity to also retrieve and show some characteristics (e.g., an image or a description in a specific language)? Cross-domain knowledge bases like DBpedia contain such properties for almost every popular entity. Moreover, how to directly integrate information coming from multiple web archives? For example, how can we combine information from a news archive and a social media archive?

Q3 *Information Inference.* How to infer knowledge by exploiting the contents of a web archive? For example, can we identify important time periods related to one or more entities? Vice-versa, can we find out the most popular entities of a specific type in a specific time period (e.g., most discussed *politicians* in articles of *2000*)? Or how can we understand the topic of a web page (e.g., find news articles related to *medicine*)?

Q4 *Robustness (in information change).* How to explore a web archive by automatically taking into account the change of entities over time? For example, the company *Accenture* was formerly known as *Andersen Consulting*, or the city *Saint Petersburg* was previously named *Leningrad*. Such temporal reference variants are common in the case of high impact events, new technologies, role changes, etc. How can we find documents from the past about such entities without having to worry about their correct reference?

Q5 *Multilinguality.* How to explore documents about entities from the past independently of the document language (and thus of the language of the entity name)? For instance, *abortion* is *Avortement* in French and *Schwangerschaftsabbruch* in German. How can we find documents about entities without having to worry about the document and entity language?

Q6 *Interoperability.* How to facilitate exploitation of web archives by other systems? How to expose information about web archives in a standard and machine understandable format, that will always be available on the Web, and that will allow for easy information integration? How to avoid downloading and parsing the entire web archive for identifying an interesting part of it related both to a time period and to some entities. For example, how can we gather a corpus of articles of *2004* discussing about *Greek politicians*?

B. Related Work

1) Profiling Web Archives: A semantic layer can be considered a way to *profile* the contents of a web archive. AlSum et al. [8] exploit the age of the archived copies and their supported domains, to avoid sending queries to archives that likely do not hold the archived page. Sawood et al. [9] examine the size and precision trade-offs in different policies for producing profiles of web archives (ranging between using full URIs and only top-level domains). Bornand et al. [10] explore the use of binary, archive-specific classifiers to determine whether or not to query an archive for a given URI. Alam et al. [11] introduce a random searcher model to randomly explore the holdings of an archive (by exploiting co-occurring terms).

The aim of all these works is to improve the effectiveness of query routing strategies in distributed archive search. However, such profiling approaches do not allow expressing semantic information about the *contents* of the archived documents and thus cannot be exploited for satisfying more sophisticated information needs like those discussed in Section II-A.

2) Exploring Web Archives: The Wayback Machine is a digital archive of the Web created by the Internet Archive (https://archive.org/). It currently contains more than 450 billion web pages, making it the biggest web archive in the world. With the Wayback Machine, the user can retrieve and access older versions of a web page. The results are displayed in a calendar view showing also the number of times the URL was crawled. Wayback Machine also offers faceted exploration of archived collections, thus allowing the user to filter the displayed results by media type, subject, collection, creator, and language. Recently, it also started offering keyword-based searching.

[2] ERC Advance Grant, Nr. 339233, http://alexandria-project.eu/.

The Portuguese Web Archive (PWA) (http://archive.pt) is a research infrastructure that enables search and access to files archived from the Web since 1996. PWA provides comprehensive crawls of the Portuguese Web and supports both keyword and URL based searching.

Memento's *Time Travel* service (http://mementoweb.org) makes it easier for users to browse the archived version of a web page by redirecting them to the archive hosting the page. The user provides the URL of the web page and a date of interest and Time Travel checks various web archives for finding an older version of the web page closest to the time indicated by the user.

Archive-It (https://archive-it.org) is a web archiving service from the Internet Archive that helps harvesting, building and preserving collections of digital content. It currently supports keyword-based searching while the user can also filter the displayed results based on several metadata values like creator, subject, and language. Padia et al. [12] present an alternative interface for exploring an Archive-It collection consisting of multiple visualizations (image plot with histogram, wordle, bubble chart and timeline).

Regarding research works, Tempas [13] is a keyword-based search system that exploits a social bookmarking service for temporally searching a web archive by indexing tags and time. It allows temporal selections for search terms, ranks documents based on their popularity and also provides query recommendations. Kanhabua et al. [14] propose a search system that uses Bing for searching the current Web and retrieving a ranked list of results. The results are then linked to the WayBack Machine thereby allowing keyword search on the Internet Archive without processing and indexing its raw contents. Vo et al. [15] study the usefulness of non-content evidences for searching web archives, where the evidences are mined only from metadata of the web pages, their links and the URLs. ArchiveWeb [16] is a search system that supports collaborative search of archived collections. It allows searching across multiple collections in conjunction with the live web, grouping of resources, and enrichment using comments and tags. Jackson et al. [17] present two prototype search interfaces for web archives. The first provides facets to filter the displayed results by several metadata values (like content type and year of crawl), while the other is a trend visualization inspired by Google's Ngram Viewer. Finally, Singh et al. [18] introduce the notion of *Historical Query Intents* and model it as a search result diversification task which intends to present the most relevant results (for free text queries) from a topic-temporal space. For retrieving and ranking historical documents (e.g., news articles), the authors propose a novel retrieval algorithm, called HistDiv, which jointly considers the aspect and time dimensions.

Although existing systems offer user-friendly interfaces, they cannot satisfy more complex (but common) information needs like those described in Section II-A. By basing upon semantic technologies, a semantic layer allows to *semantically* describe the contents of a web archive and to directly "connect" them with existing information available on online knowledge bases (like DBpedia). In that way, we can not only explore archived documents in a more advanced way, but also integrate information, infer new knowledge and quickly identify interesting parts of a web archive for further analysis.

The main drawback of our approach is its user-friendliness since, currently, for querying a semantic layer one has to write structured (SPARQL) queries. However, user-friendly interfaces can be developed on top of semantic layers that will allow end-users to easily explore them. Moreover, we can directly exploit systems like Sparklis [19] that allow to explore the contents of semantic repositories through a Faceted Search-like interface [20], [21]. There are also approaches that translate free-text queries to SPARQL (like [22]). Providing such user-friendly interfaces on top of semantic layers is out of the scope of this paper but an important direction for future research.

3) Analyzing Web Archives: Lin et. al. [23] propose a platform for analyzing web archives, called Warcbase, which is built on Apache HBase, a distributed data store. Storing the data using HBase allows the use of tools in the Hadoop ecosystem for efficient analytics and data processing. Warcbase also provides browsing capabilities similar to the Wayback Machine allowing users to access historical versions of captured web pages.

ArchiveSpark [24] is a programming framework for efficient and distributed web archive processing. It is based on the Apache Spark cluster computing framework [25] and makes use of standardized data formats for analyzing web archives. *ArchiveSpark2Triples* is an extension of ArchiveSpark for efficiently creating semantic layers for web archives (more in Section III-C).

III. Building Semantic Layers

A. The "Open Web Archive" Data Model

We first introduce an RDF/S data model for describing metadata and semantic information about the documents of a web archive. Figure 1 depicts the proposed model, which we call *Open Web Archive* data model.[3] We have defined 2 new classes and 3 new properties, while we exploit elements from many other data models. The class `owa:ArchivedDocument` represents a document that has been archived. An archived document may be linked or may not be linked with some versions, i.e., instances of `owa:VersionedDocument`. For example, an archived article from the New York Times corpus [26] does not contain versions. On the contrary, Internet Archive contains versions for billions of web sites. For the case of versioned web archives, and with correspondence to the Memento Framework (RFC 7089) [27], an archived document corresponds to an *Original Resource* and a versioned document to a *Memento*. An archived document containing versions can be also associated with some metadata information like the date of its first capture (using the property `owa:firstCapture`), the date of its last capture (using the property `owa:lastCapture`) as well as its total number of captures (using the property `owa:numOfCaptures`).

An archived or versioned document can be associated with three main kinds of elements: i) with metadata information like date of publication/capture, title of document, and format (mime type), ii) with other archived or not documents (i.e., links to other web pages), and iii) with a set of annotations.

[3] The specification is available at: http://l3s.de/owa/

For describing some of the metadata we exploit terms of the Dublin Core Metadata Initiative[4]. For describing an annotation, we exploit the Open Annotation Data Model [28] and the Open Named Entity Extraction (NEE) Model [29]. The Open Annotation Data Model specifies an RDF-based framework for creating associations (annotations) between related resources, while the Open NEE Model is an extension that allows describing the result of an entity extraction process. An annotation has a *target*, which in our case is an archived or versioned document, and a *body* which is an entity mentioned in the document. We can also directly relate an archived or versioned document with an entity by exploiting the property "*mentions*" of schema.org[5]. This can highly reduce the number of derived triples. An entity can be associated with information like its name, a confidence score, its position in the document, and a resource (URI). The URI enables to retrieve additional information from the Linked Open Data (LOD) cloud [30] (like properties and relations with other entities).

Figure 2 depicts an example of an archived non-versioned article. We can see some of its metadata values (date, format, title), its references to other web pages, and its annotations. We notice that the entity name "Federer" was identified in that document. We can also see that this entity has been linked with the DBpedia resource corresponding to the tennis player *Roger Federer*. By accessing DBpedia, we can now retrieve more information about this entity like its birth date, an image, a description in a specific language, etc. Such links to DBpedia can also take the temporal aspect into account. For example, we can provide entity URIs that lead to DBpedia entity descriptions as they were at the time the web page was captured (e.g., by exploiting DBpedia archives provided by Memento[6]).

Figure 3 depicts an example of an archived web page containing versions. Now, each version has its own metadata, annotations and references to other web pages. We notice that the event name "Euro 2008" was identified in the first version of the archived document and was linked to the DBpedia resource corresponding to the soccer tournament *UEFA Euro 2008*. The archived document is also associated with metadata information related to its versions. Specifically we can see the date of its first capture, the date of its last capture and its total number of captures. In addition, by exploiting the *same-as* property of OWL Web ontology language [31], we can define that a specific version of a URL is the same as another version (e.g., versions 2 and 3 in our example). Thereby, we can avoid storing exactly the same data for two identical versions (redundancy is a common problem in web archives).

Extensibility and Update. The proposed model is highly extensible. For instance, we can exploit the VoID Vocabulary [32] and express dataset-related information like statistics (number of triples, number of entities, etc.), creation or last modification date, the subject of the dataset, and collection from which the dataset was derived. Likewise, one may exploit the PROV data model [33] and store provenance-related information (e.g., which tool was used for crawling the documents or for annotating them, what organizations or people were involved in the crawling or annotation process, etc.). In addition, since the contents of the archived documents never change, we can easily update a semantic layer by just adding triples in the RDF repository, e.g., for describing more metadata about the archived documents or for including new versions. In the latter case only, we should also update the date of last capture and the total number of captures of the corresponding archived document.

B. The Construction Process

For constructing a semantic layer we follow the following process:

- *Read/Extraction of main content and metadata*. We first extract the main content (full text) from each archived document (for annotating it with entities) and we also read its metadata. This, of course, depends on the format used for storing the archive. For example, WARC (ISO 28500:2009) is the standard format for storing web crawls, CDX is widely used for describing metadata of web documents, while NITF (News Industry Text Format) is a standard XML-based format for storing and sharing news articles. For extracting the main content from HTML web pages, we should also remove the surplus around the main textual content of a web page (boilerplate, templates, etc.). We can also extract any other information related to an archived document that we may want to semantically describe, like the title of the web page or links to other web pages.

- *Entity extraction and linking*. We apply entity extraction and linking in the full text of each archived document for detecting entities, events and concepts mentioned in the document and associating them with web resources (like DBpedia/Wikipedia URIs). TagMe [34], AIDA [35] and BabelFy [36] are well-known entity extraction and linking tools with satisfactory performance in entity disambiguation.

- *Schema-based generation of RDF triples*. Now, we exploit the *Open Web Archive* data model, as well as any other needed vocabulary/ontology, for generating the RDF triples that describe all the desired data related to the archived documents (metadata, entities, etc.). For representing the extracted entities (instances of `oa:Annotation`, `oae:Entity`, `dc:Event`, and `dc:Concept`), we can use blank nodes [37] (since such information does not need to be assigned a unique URI). We can use blank nodes for also naming the archived or versioned documents (instances of `owa:ArchivedDocument` or `owa:VersionedDocument`) in case no URLs are given by the archive provider and no other URLs can be used (e.g., links to the Wayback Machine). Moreover, for the case of versioned documents, if a specific version of a document is the same as an older version of the same document (e.g., in case they have the same checksum), we can add a *same-as* link starting from the newer document and pointing to the older one (thereby avoiding storing identical information).

- *Entity enrichment (optionally)*. We can enrich the extracted entities with more information coming from other knowledge bases (like properties, characteristics and relations with other entities). The LOD cloud contains hundreds of knowledge bases covering many domains. In that way the semantic layer can directly offer more data about the extracted entities, allow-

[4] http://dublincore.org/

[5] http://schema.org/mentions

[6] http://mementoweb.org/depot/native/dbpedia/

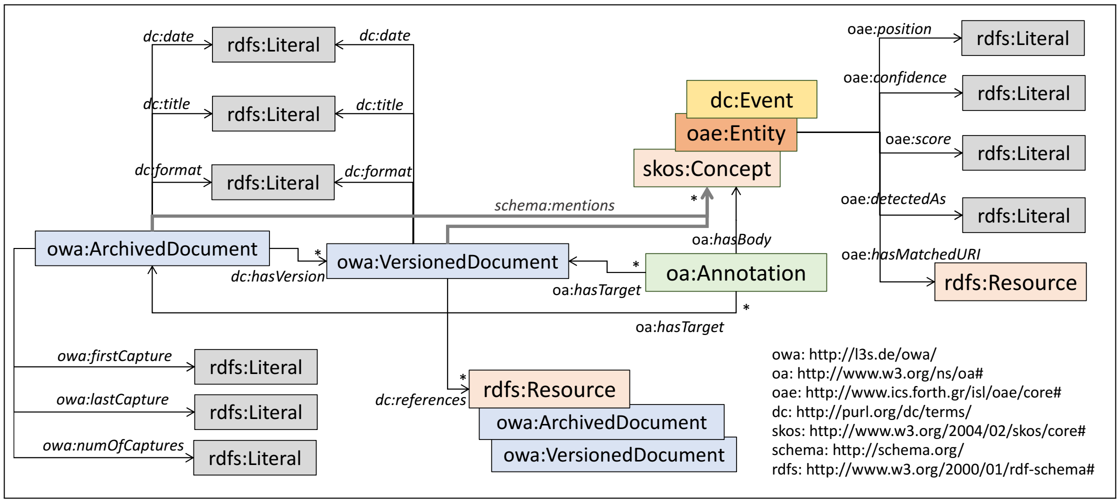

Fig. 1: The *Open Web Archive* data model.

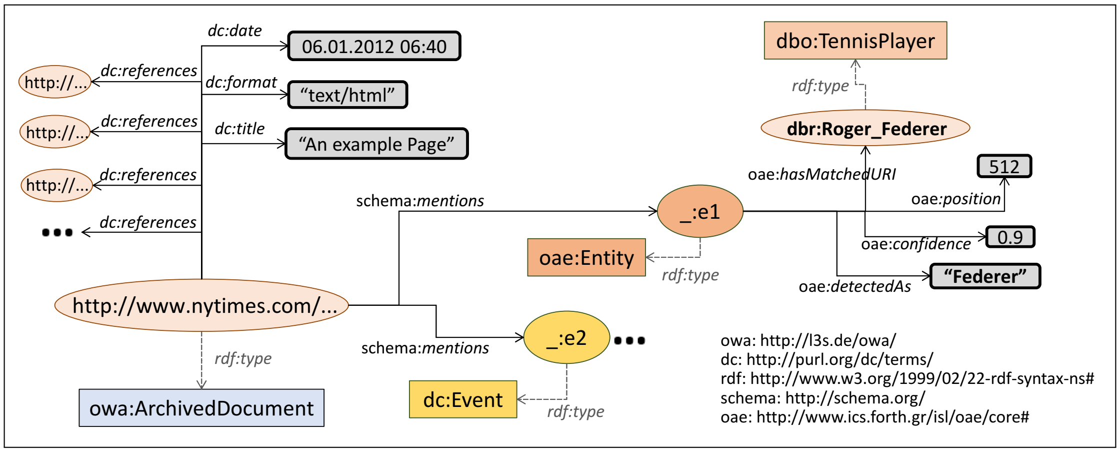

Fig. 2: Describing an archived article (non-versioned) using the *Open Web Archive* data model.

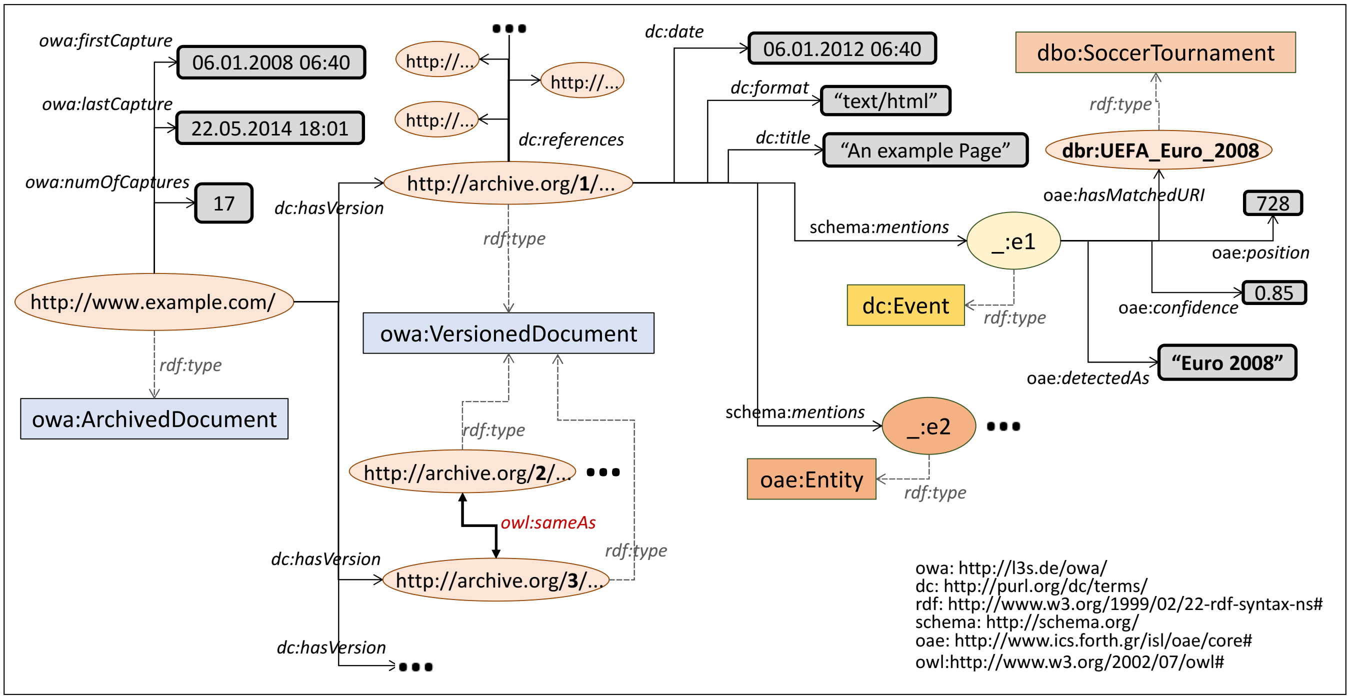

Fig. 3: Describing an archived web page (versioned) using the *Open Web Archive* data model.

ing for more sophisticated query capabilities and faster query answering, without requiring to access external knowledge bases. This step can be also performed after the construction of the semantic layer, at any time, since we just have to add triples describing information about the entities in the repository.

- *Storage*. The derived RDF triples are stored in a triplestore (e.g., OpenLink Virtuoso[7]). Now, we can access the triplestore and query the semantic layer through SPARQL.

- *Publication (optionally)*. We can make the triplestore publicly available through a SPARQL endpoint and/or as Linked Data. This will allow other applications to directly access and query the semantic layer.

C. The "ArchiveSpark2Triples" Framework

ArchiveSpark [24] is a programming framework for efficiently analyzing web archives stored in the standard WARC/ CDX format. The core of ArchiveSpark is its unified data model which stores records in an hierarchical way, starting with the most essential metadata of a webpage like its URL, timestamp, etc. Based on this metadata, ArchiveSpark can run basic operations such as filtering, grouping and sorting very efficiently. In a step-wise approach the records can be enriched with more information by applying external modules, called *enrich functions*. An *enrich function* can call any third-party tool to extract or generate new information from the contents of a web page. These functions can be fully customized and shared among researchers and tasks.

ArchiveSpark2Triples[8] is an extension of ArchiveSpark that automates the construction of a semantic layer. It reads a web archive and outputs information about its resources as well as derived information in the Notation3 (N3) RDF format based on the *Open Web Archive* data model. Internally, *ArchiveSpark2Triples* defines three types of documents: *archived document* (instance of `owa:ArchivedDocument`), *versioned document* (instance of `owa:VersionedDocument`), and *same-as versioned document* (instance of `owa:VersionedDocument` which constitutes a *revisit-record*, i.e., duplicate of a previous capture). In more detail:

- An *archived document* represents all versions of the same web page, i.e., all records with the same URL. Its triples reflect the web page as one unit, including the number of captures in the web archive, the timestamps of the first and last capture as well as pointers to the corresponding *versioned documents*.
- A *versioned document* represents each capture of a web page, i.e., every record of a web page in the archive. The assignment of URLs to the versioned documents is customizable and thus can be defined by the user. By default, the triples of such a document only include the date of the capture and its mime type (e.g., text, image, etc.). However, the framework supports to extend this easily by accessing and transforming into triples any property of ArchiveSpark's data model. If this step involves *enrich functions*, the required content of the web page is seamlessly integrated by ArchiveSpark's enrichment mechanisms. In our case, we can use enrich functions to extract the title of a page, its links to other pages, and its entities. The extraction of entities requires an additional module which uses the entity extraction and linking system Yahoo FEL [38]. The corresponding enrich function is available under *FEL4ArchiveSpark*[9].
- A *same-as versioned document* represents an already archived web page whose content has not been changed. In this case, a *same-as* property pointing to the previous record is only created. The way in which duplicates are identified is not part of the framework and can be defined as part of the generation workflow.

Finally, defining the vocabularies to use for producing the triples is part of the generation workflow and thus can be customized by the user. A Jupyter Notebook with an example of a workflow is publicly available[10].

Efficiency. *ArchiveSpark2Triples* gains its efficiency from the efficiency of ArchiveSpark, which is mainly a result of the two-way approach that is used for data loading and access [24]. An archived collection to be used with ArchiveSpark always consists of two parts, the WARC files containing the data records with headers and payloads, and the CDX files containing only basic metadata such as URLs, timestamps and datatype (which are considerably smaller in size). Hence, operations that rely exclusively on information contained in the metadata can be performed very efficiently, e.g., filtering out items of a certain type. Eventually, if operations need to be performed on the actual contents, only the required records are accessed using location pointers in the CDX files. *ArchiveSpark2Triples* benefits from this approach, since records of a datatype other than *text/html*, such as images and videos, can be filtered out very fast. In addition, all properties of the *archived documents* and the majority of properties of the *versioned documents* can be generated purely on metadata and thus, very efficiently. In fact, the payload is accessed only for applying *enrich functions*, e.g., for extracting the title of a web page, its entities, etc. However, these are only part of the *same-as versioned documents* that do not constitute duplicates.

The most expensive task in our pipeline is the entity extraction process, performed by *FEL4ArchiveSpark* using Yahoo FEL [38] (a lightweight and very efficient entity extraction and linking system). To avoid extraordinarily long runtimes, *FEL4ArchiveSpark* supports to define a timeout (e.g., set to 10 seconds per record in our experiments). Additionally, we consider only web pages with a compressed size of less than 100 KB, as larger file sizes are unlikely to constitute a web page and may indicate a malformed record. Although the described steps are considered quite efficient, the actual time for the entire workflow depends on the dataset size, the nature of the data as well as the used computing infrastructure. Indicatively, the Hadoop cluster used in our experiments for producing a semantic layer for a web archive of about 9 millions web pages consisted of 25 compute nodes with a total of 268 CPU cores and 2,688 GB RAM (more about this web archive in Section IV-A). While the available resources strongly depend on the load of the cluster and vary, we worked

[7] https://virtuoso.openlinksw.com/

[8] https://github.com/helgeho/ArchiveSpark2Triples

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

[10] https://github.com/helgeho/ArchiveSpark2Triples/blob/master/notebooks/Triples.ipynb

with 110 executors in parallel most of the time, which resulted in a runtime of 24 hours for processing the entire collection of 474.6 GB of compressed WARC and CDX files.

IV. Case Studies

In this section, we present three semantic layers for three different types of web archives and we showcase their query capabilities. The semantic layers are publicly available for experimentation and further research[11].

A. A Semantic Layer for a Web Archive

Using *ArchiveSpark2Triples*, we created a semantic layer for the *Occupy Movement 2011/2012* collection[12], which has been generously provided to us by Archive-It. The collection contains 9,094,573 captures of 3,036,326 web pages related to protests and demonstrations around the world calling for social and economic equality. For each version, we stored its *capture date*, its *title*, its *mime type* and its *extracted entities* (using a confidence score of -4), while for each distinct URL we stored its total number of captures, the date of its first capture, and the date of its last capture. For assigning URLs to the versioned web pages, we used links to the collection's Wayback Machine provided by Archive-It. In that way one can have direct online access to a specific version of an archived web page.

The semantic layer contains 1,344,450 *same-as* properties, which means that we avoided annotating and storing identical information for a very large number of versioned web pages. Moreover, 939,960 distinct entities (including concepts and events) were extracted from the archived web pages. For each entity, we stored its name (surface form), its URI, its position in the text, and its confidence score. The constructed semantic layer contains totally more than 10 billion triples (10,884,509,868).

B. A Semantic Layer for a News Archive

We created a semantic layer for the New York Times (NYT) Annotated Corpus [26] (a non-versioned news archive). The corpus contains over 1.8 million articles published by the NYT between 1987 and 2007. We filtered out articles like memorial notices, corrections, letters, captions, etc. which actually are not articles. This reduced their number to 1,456,896. For each article in the corpus, a large amount of metadata is provided. In this case study, we exploited only the article's *URL*, *title* and *publication date*. Of course, one can exploit any other of the provided metadata (like *author*, *taxonomic classifiers*, etc.) and extend the semantic layer with more triples describing these metadata fields.

We used TagMe [34] for extracting entities from each article using a confidence score of 0.2. For each extracted entity, we stored its name (surface form), its URI and its confidence score. In total, 856,283 distinct entities (including concepts and events) were extracted from the NYT articles. The constructed semantic layer contains totally 195,958,390 triples.

C. A Semantic Layer for a Social Media Archive

We also created a semantic layer for a collection of tweets. The collection comprises 1,363,487 tweets posted in 2016 by 469 twitter accounts of USA newspapers. For each tweet we exploit its *text*, *creation date*, *favorite count*, *retweet count*, and the *screen name* of the account that posted the tweet. For representing an instance of a tweet, as well as its favorite and retweet count, we used the OpenLink Twitter Ontology[13] (its class `Tweet` corresponds to an *archived document* in our model).

For extracting entities from the tweets, we used Yahoo FEL (with confidence score -4). For each extracted entity, we stored its name (surface form), its URI and its confidence score. In total, 146,854 distinct entities (including concepts and events) were extracted from the collection. The constructed semantic layer contains totally 19,242,761 triples.

D. Query capabilities

By exploiting the expressive power of SPARQL and its federated features, we can offer advanced query capabilities over the semantic layers. Below we discuss how a semantic layer can satisfy the motivating questions described in Section II-A by also presenting interesting query examples.

Information Exploration (Q1) and Integration (Q2). A semantic layer allows running sophisticated queries that can also directly integrate information from external knowledge bases. For example, Figure 4 shows a SPARQL query that can be answered by the semantic layer of the NYT corpus. The query asks for articles of June 1989 discussing about New York lawyers born in Brooklyn. By directly accessing DBpedia, the query retrieves the entities that satisfy the query as well as additional information (in our example the birth date and a description in French of each lawyer). The query returns 47 articles mentioning 5 different New York lawyers born in Brooklyn.

```
SELECT ?article ?title ?date ?nylawyer ?bdate ?abstr WHERE {
 SERVICE <http://dbpedia.org/sparql> {
  ?nylawyer dc:subject dbc:New_York_lawyers ;
            dbo:birthPlace dbr:Brooklyn .
  OPTIONAL {
   ?nylawyer dbo:birthDate ?bdate ;
             dbo:abstract ?abstr FILTER(lang(?abstr)="fr")}}
 ?article dc:date ?date FILTER(?date>="1989-06-01"^^xsd:date
                        && ?date<="1989-06-30"^^xsd:date)
 ?article schema:mentions ?entity .
 ?entity oae:hasMatchedURI  ?nylawyer .
 ?article dc:title ?title } ORDER BY ?nylawyer
```

Fig. 4: SPARQL query for retrieving articles of June 1989 discussing about New York lawyers born in Brooklyn.

Figure 5 shows a query that can be answered by the semantic layer of the tweets collection. The query requests the most popular tweets (having more than 50 retweets) posted during the summer of 2016, mentioning basketball players of the NBA team *Los Angeles Lakers*. The query returns 14 tweets mentioning 7 different players.

We can also combine information coming from different semantic layers. As an example, one could run a query requesting tweets of 2016 mentioning basketball players discussed in articles of the same time period.

[11] http://l3s.de/owa/semanticlayers/

[12] https://archive-it.org/collections/2950

[13] http://www.openlinksw.com/schemas/twitter

```
SELECT DISTINCT ?tweet ?count ?date ?entityUri WHERE {
 SERVICE <http://dbpedia.org/sparql> {
   ?entityUri dc:subject dbc:Los_Angeles_Lakers_players }
 ?t a tw:Tweet ;
   dc:date ?date FILTER(?date>="2016-06-01"^^xsd:dateTime &&
                       ?date<="2016-08-31"^^xsd:dateTime)
 ?t tw:retweetCount ?count FILTER (?count > 50) .
 ?t schema:text ?tweet ; schema:mentions ?entity .
 ?entity oae:hasMatchedURI ?entityUri }
```

Fig. 5: SPARQL query for retrieving popular tweets about basketball players of Los Angeles Lakers.

Information Inference (Q3). By querying a semantic layer we can infer useful knowledge related to the archived documents that is very laborious to derive otherwise. For example, Figure 6 shows a query that can be answered by the semantic layer of the *Occupy Movement* collection. The query asks for the most discussed journalists in the web pages of this collection. Notice that the query counts the archived documents, not the versions. In that way we avoid counting multiple times exactly the same pages captured in different time periods. The query returns *Ralph Nader*, *Chris Hedges* and *Dylan Ratigan*, as three of the most discussed journalists.

```
SELECT ?journ (COUNT(DISTINCT ?page) AS ?num) WHERE {
 SERVICE <http://dbpedia.org/sparql> {
   ?journ a yago:Journalist110224578 }
 ?page a owa:ArchivedDocument ; dc:hasVersion ?version .
 ?version schema:mentions ?entity .
 ?entity oae:hasMatchedURI  ?journ .
} GROUP BY ?journ ORDER BY DESC(?num)
```

Fig. 6: SPARQL query for retrieving the most discussed journalists in web pages of the *Occupy Movement* collection.

Likewise, by running a query at the semantic layer of the NYT corpus requesting the number of articles per year discussing about *Nelson Mandela* (Figure 7), we can see that in 1990 the number of articles is much higher compared to the previous years, meaning that this year was probably important for *Nelson Mandela* (indeed, as in 1990 *Nelson Mandela* was released from prison).

```
SELECT ?year (COUNT(DISTINCT ?article) AS ?num) WHERE {
  ?article dc:date ?date ; schema:mentions ?entity .
  ?entity oae:hasMatchedURI dbr:Nelson_Mandela
} GROUP BY (year(?date) AS ?year) order by ?year
```

Fig. 7: SPARQL query for retrieving the number of articles per year mentioning *Nelson Mandela*.

Robustness (Q4) and Multilinguality (Q5). Each entity extracted from the archived documents is assigned a unique URI (together with a confidence score) which can be used for retrieving documents and information related to that entity. This means that all different mentions of an entity (e.g., name variants or names in different languages) are assigned the same unique URI. Thereby, we can query a semantic layer and retrieve information related to one or more entities without having to worry about the names of the entities (like in the queries of Figures 4-6). Of course, this also depends on the entity linking system used for extracting the entities, specifically on its "time-awareness" and correct disambiguation (e.g., for understanding that *Leningrad* corresponds to the DBpedia URI http://dbpedia.org/resource/Saint_Petersburg), as well as on whether it supports the identification of entities in different languages (e.g., for assigning the same URI http://dbpedia.org/resource/Abortion to both "abortion" and "Schwangerschaftsabbruch").

Interoperability (Q6). RDF is a standard model for data interchange on the Web and has features that facilitate data integration. Describing metadata and content-based information about web archives in RDF makes their contents machine *understandable*, and allows their direct exploitation by other systems and tools. Moreover, following the LOD principles for publishing a semantic layer enables other systems to directly access it, while the advanced query capabilities that it offers allow the easy identification of an interesting part of a web archive (related to a time period and some entities) by just writing and submitting a SPARQL query.

V. Evaluation

Our objective is to show that for a bit more complex information needs (e.g., of exploratory nature), keyword-based search systems return poor results and thus there is the need for more advanced information seeking strategies. This corresponds to our first motivating questions (**Q1**). We also study the quality of the results returned by a semantic layer for identifying possible problems and limitations.

Setup. We have defined a set of 20 information needs of exploratory nature. Each information need requests documents of a specific time period, related to some entities of interest. We used the NYT corpus as the underlying archived collection. For example, *"find articles of June 2010 discussing about African-American film producers"* is such an *exploratory* information need.

Each of the information needs corresponds to a SPARQL query and to a free-text query that better describes the information need (in our evaluation we consider one interaction step, i.e., one submitted query). As an example, for the information need *"find articles of June 2010 discussing about African-American film producers"*, the free-text query that is used is *"African-American film producer"* (we manually specify the date range to each system). We evaluated and compared the results returned by the SPARQL query over the semantic layer with the results returned by the following two keyword-based search systems operating over the NYT corpus: a) Google News (adding at the end of the query the string "site:nytimes.com" for returning only results from this domain), b) HistDiv [18] (which uses a different, diversity-oriented approach for searching news archives). Moreover, in the reported results we did not consider 23 articles (out of totally 356 articles) returned by the SPARQL queries because they do not exist in Google News.

For each information need, we measure: i) the number of hits returned by the SPARQL query (denoted as S); ii) the number of *relevant* hits returned by the SPARQL query (denoted as S_R); iii) the number of hits returned by each search system (denoted as K); iv) the number of *relevant* hits returned by each search system, existing in the set of relevant hits returned by the SPARQL query (denoted as $K_R \in S_R$); and v) The number of *relevant* hits returned by each search system, not existing in the set of relevant hits returned by the SPARQL query (denoted as $K_R \notin S_R$). The

TABLE I: Comparative evaluation results on effectiveness.

#	S	S_R	Google			HistDiv		
			K	$K_R \in S_R$	$K_R \notin S_R$	K	$K_R \in S_R$	$K_R \notin S_R$
1	27	27	8	0	0	0	0	0
2	34	27	1	0	1	3	2	1
3	37	33	0	0	0	1	0	0
4	16	16	0	0	0	0	0	0
5	11	9	0	0	0	0	0	0
6	14	14	1	0	0	0	0	0
7	18	2	1	0	0	0	0	0
8	8	8	1	0	0	4	0	3
9	11	1	0	0	0	0	0	0
10	15	14	0	0	0	0	0	0
11	15	1	0	0	0	0	0	0
12	12	8	0	0	0	0	0	0
13	13	13	0	0	0	0	0	0
14	16	15	2	0	0	0	0	0
15	15	9	0	0	0	0	0	0
16	12	10	6	0	0	25	3	3
17	15	13	1	1	0	2	1	0
18	13	11	1	0	0	0	0	0
19	16	15	1	0	0	0	0	0
20	15	15	1	0	1	0	0	0

set of information needs (together with the corresponding SPARQL and free-text queries), as well as the full results and the relevance judgements, are publicly available[14].

Results. Table I shows the results. We notice that the keyword-based search systems cannot retrieve many relevant hits, while for many cases the number of returned results is zero. This illustrates that their effectiveness is poor for more advanced information needs like those in our experiments (considering however that we allow one interaction step). The reason for this poor performance is the fact that each information need describes a category of entities which refers to a number of (possibly unknown) entities, while the corresponding free-text query does not contain the entity names. For example, the query *"African-American film producer"* does not contain the actual names of any of these film producers. Note that during an exploratory search process, users may be unfamiliar with the domain of their goal (e.g., they may not know the names of the entities of interest), may be unsure about the ways to achieve their goal (e.g., not sure about the query to submit to a search system), or may need to learn about the topic in order to understand how to achieve their goal (e.g., learn facts about some entities of interest) [2]. For achieving a better performance, the user should probably first find entities belonging to the corresponding information need and then submit queries using the entity names in the query terms. Thus, multiple interaction and exploration steps may be needed. However this can be infeasible, for example in case of a large number of entities of interest.

Nevertheless, the results also show that in a few cases the search system returns relevant hits that are not returned by the SPARQL query (e.g., #2 and #20 for Google, #2, #8 and #16 for HistDiv). In addition, some of the hits returned by the SPARQL query are not relevant (e.g., 7 results of #2), while especially in three cases (#7, #9, and #11), this number is very large. This is due to disambiguation error of the entity linking system. For example, for the information need #9 (*"Find articles discussing about Australian Cricketers who played One Day Internationals"*), the entity extraction system wrongly linked the name "John Dyson" to the former international cricketer John Dyson, instead of the deputy mayor John Dyson (at the time of Rudolph Giuliani's mayoralty) discussed in the articles. Therefore, the performance of the entity extraction system as well as the confidence threshold used for entity disambiguation can affect the quality of the retrieved results. Applying a low confidence threshold can increase recall, however many irrelevant hits may also be returned. On the contrary, by applying a high confidence threshold, the returned results are less but the probability that they are correct is higher.

To sum up, we have identified the following problems that can affect the quality of the results:
– *False positive:* A SPARQL query may return a result which is not relevant, due to disambiguation error of the underlying entity extraction system.
– *False negative:* A SPARQL query may not return a relevant result because: i) the entity extraction system did not manage to recognize one of the entities of interest, ii) the entity extraction system did not disambiguate correctly an extracted entity of interest, iii) the confidence score of the extracted entity of interest is under the threshold used for entity disambiguation.
– *Temporal inconsistency:* A SPARQL query may return an irrelevant hit or may not return a relevant hit, because a property of an entity of interest has changed value. For example, the query of Figure 5 may return a tweet for a basketball player who was playing in a different team at the time the tweet was posted (although this also depends on user's intention, since he/she may be interested in also such players). Likewise, a query may not return a hit because the knowledge base (from which we retrieve the list of players) may not contain information about the team's old players. Thus, the contents of the knowledge base, its "freshness" and its completeness, affect the quality of the retrieved results.

Efficiency of Query Answering: The execution time of a SPARQL query over a semantic layer mainly depends on: a) the efficiency of the triplestore hosting the semantic layer (e.g., in-memory triplestores are more efficient), b) the efficiency of the server hosting the triplestore (available main memory, etc.), and c) the query itself since some SPARQL operators are costly (like the operators `FILTER` and `OPTIONAL`). Moreover, if the query contains one or more `SERVICE` operators (like the queries of Figures 4-6), then its execution time is also affected by the efficiency of the remote endpoints at the time of the request.

Indicatively, the average execution time of the 20 queries used in our evaluation was about 400 ms, with minimum 56 ms for #16 and maximum 2.4 sec for #15 (we run the queries 10 times within 3 days and here we report the average values). All these queries use the `SERVICE` operator for querying DBpedia's SPARQL endpoint but not any `FILTER` or `OPTIONAL` operator, while the semantic layer was hosted in a Virtuoso server installed in a modest personal computer (MacBook Pro, Intel Core i5, 8GB main memory) and we run the queries in Java 1.8 using Apache Jena 3.1.

[14] http://l3s.de/owa/semanticlayers/SemLayerEval.zip

VI. Conclusion

We have introduced a model and a framework for describing and publishing metadata and semantic information about web archives. The constructed *semantic layers* allow: i) exploring web archives in a more advanced way based on entities, events and concepts extracted from the archived documents and linked to web resources; ii) integrating information (even at query-execution time) coming from multiple knowledge bases and semantic layers; iii) inferring new knowledge that is very laborious to derive otherwise; iv) coping with common problems when exploring web archives like temporal reference variants and multilinguality; and v) making the contents of web archives machine understandable, thereby enabling their direct exploitation by other systems and tools. The results of a comparative evaluation showed that semantic layers can answer complex information needs that keyword-based search systems fail to satisfy. The evaluation also enabled us to identify problems that can affect the effectiveness of query answering.

We believe that constructing semantic layers is the first step towards more advanced and meaningful exploration of web archives. Our vision is to enrich the LOD cloud[15] with semantic layers, i.e., with knowledge bases describing metadata and semantic information about archived collections.

Regarding future work and research, user-friendly interfaces should be developed on top of semantic layers for allowing end-users to easily and efficiently explore web archives. Another interesting direction is to study approaches for ranking the results returned by SPARQL queries.

Acknowledgements. The work was partially funded by the European Commission for the ERC Advanced Grant ALEXANDRIA (No. 339233).

References

[1] G. Weikum, M. Spaniol, N. Ntarmos, P. Triantafillou, A. Benczúr, S. Kirkpatrick, P. Rigaux, and M. Williamson, “Longitudinal analytics on web archive data: It’s about time!” in *5th Biennial Conference on Innovative Data Systems Research*. CIDR 2011, 2011.

[2] G. Marchionini, “Exploratory search: from finding to understanding,” *Communications of the ACM*, vol. 49, no. 4, 2006.

[3] M. Whitelaw, “Generous interfaces for digital cultural collections,” *Digital Humanities Quarterly*, vol. 9, no. 1, 2015.

[4] D. Brickley, R. V. Guha, and B. McBride, “Rdf schema 1.1,” *W3C recommendation*, 2014.

[5] E. PrudHommeaux, A. Seaborne *et al.*, “Sparql query language for rdf,” *W3C recommendation*, vol. 15, 2008.

[6] E. Prudhommeaux, C. Buil-Aranda *et al.*, “Sparql 1.1 federated query,” *W3C Recommendation*, vol. 21, 2013.

[7] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer *et al.*, “Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia,” *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.

[8] A. AlSum, M. C. Weigle, M. L. Nelson, and H. Van de Sompel, “Profiling web archive coverage for top-level domain and content language,” *International Journal on Digital Libraries*, vol. 14, no. 3-4, pp. 149–166, 2014.

[9] S. Alam, M. L. Nelson, H. Van de Sompel, L. L. Balakireva, H. Shankar, and D. S. Rosenthal, “Web archive profiling through cdx summarization,” in *International Conference on Theory and Practice of Digital Libraries*. Springer, 2015.

[10] N. J. Bornand, L. Balakireva, and H. Van de Sompel, “Routing memento requests using binary classifiers,” in *16th ACM/IEEE-CS on Joint Conference on Digital Libraries*. ACM, 2016.

[11] S. Alam, M. L. Nelson, H. Van de Sompel, and D. S. Rosenthal, “Web archive profiling through fulltext search,” in *International Conference on Theory and Practice of Digital Libraries*. Springer, 2016.

[12] K. Padia, Y. AlNoamany, and M. C. Weigle, “Visualizing digital collections at archive-it,” in *12th ACM/IEEE-CS joint conference on Digital Libraries*. ACM, 2012.

[13] H. Holzmann and A. Anand, “Tempas: Temporal archive search based on tags,” in *International Conference on World Wide Web*, 2016.

[14] N. Kanhabua, P. Kemkes, W. Nejdl, T. N. Nguyen, F. Reis, and N. K. Tran, “How to search the internet archive without indexing it,” in *20th International Conference on Theory and Practice of Digital Libraries*. Springer, 2016.

[15] K. D. Vo, T. Tran, T. N. Nguyen, X. Zhu, and W. Nejdl, “Can we find documents in web archives without knowing their contents?” in *ACM Conference on Web Science*, 2016.

[16] Z. T. Fernando, I. Marenzi, W. Nejdl, and R. Kalyani, “Archiveweb: Collaboratively extending and exploring web archive collections,” in *International Conference on Theory and Practice of Digital Libraries*. Springer, 2016.

[17] A. Jackson, J. Lin, I. Milligan, and N. Ruest, “Desiderata for exploratory search interfaces to web archives in support of scholarly activities,” in *16th ACM/IEEE-CS on Joint Conference on Digital Libraries*. ACM, 2016.

[18] J. Singh, W. Nejdl, and A. Anand, “History by diversity: Helping historians search news archives,” in *ACM Conference on Human Information Interaction and Retrieval*, 2016.

[19] S. Ferré, “Sparklis: a sparql endpoint explorer for expressive question answering,” in *ISWC Posters & Demonstrations Track*, 2014.

[20] G. M. Sacco and Y. Tzitzikas, *Dynamic taxonomies and faceted search: theory, practice, and experience*. Springer Science & Business Media, 2009, vol. 25.

[21] Y. Tzitzikas, N. Manolis, and P. Papadakos, “Faceted exploration of rdf/s datasets: a survey,” *Journal of Intelligent Information Systems*, pp. 1–36, 2016.

[22] C. Unger, L. Bühmann, J. Lehmann, A.-C. Ngonga Ngomo, D. Gerber, and P. Cimiano, “Template-based question answering over rdf data,” in *21st international conference on World Wide Web*. ACM, 2012.

[23] J. Lin, M. Gholami, and J. Rao, “Infrastructure for supporting exploration and discovery in web archives,” in *International Conference on World Wide Web*, 2014.

[24] H. Holzmann, V. Goel, and A. Anand, “Archivespark: Efficient web archive access, extraction and derivation,” in *16th ACM/IEEE-CS on Joint Conference on Digital Libraries*. ACM, 2016.

[25] “Apache spark: Lightning-fast cluster computing,” 2015.

[26] E. Sandhaus, “The new york times annotated corpus,” *Linguistic Data Consortium, Philadelphia*, vol. 6, no. 12, 2008.

[27] H. Van de Sompel, M. Nelson, and R. Sanderson, “Rfc 7089-http framework for time-based access to resource states-memento,” *Internet Engineering Task Force (IETF), RFC*, 2013.

[28] R. Sanderson, P. Ciccarese, H. Van de Sompel, S. Bradshaw, D. Brickley, L. J. G. a Castro, T. Clark, T. Cole, P. Desenne, A. Gerber *et al.*, “Open annotation data model,” *W3C community draft*, 2013.

[29] P. Fafalios, M. Baritakis, and Y. Tzitzikas, “Exploiting linked data for open and configurable named entity extraction,” *International Journal on Artificial Intelligence Tools*, vol. 24, no. 02, 2015.

[30] T. Heath and C. Bizer, “Linked data: Evolving the web into a global data space,” *Synthesis lectures on the semantic web: theory and technology*, vol. 1, no. 1, pp. 1–136, 2011.

[31] S. Bechhofer, “Owl: Web ontology language,” in *Encyclopedia of Database Systems*. Springer, 2009, pp. 2008–2009.

[32] K. Alexander, R. Cyganiak, M. Hausenblas, and J. Zhao, “Describing linked datasets with the void vocabulary,” 2011.

[33] L. Moreau and P. Missier, “The prov data model,” 2013.

[34] P. Ferragina and U. Scaiella, “Tagme: on-the-fly annotation of short text fragments (by wikipedia entities),” in *19th ACM international conference on Information and knowledge management*. ACM, 2010.

[35] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum, “Robust disambiguation of named entities in text,” in *Conference on Empirical Methods in Natural Language Processing*, 2011.

[36] A. Moro, A. Raganato, and R. Navigli, “Entity linking meets word sense disambiguation: a unified approach,” *Transactions of the Association for Computational Linguistics*, vol. 2, 2014.

[37] D. Beckett and B. McBride, “Rdf/xml syntax specification (revised),” *W3C recommendation*, vol. 10, 2004.

[38] R. Blanco, G. Ottaviano, and E. Meij, “Fast and space-efficient entity linking in queries,” in *Eight ACM International Conference on Web Search and Data Mining*. New York, NY, USA: ACM, 2015.

[15] http://lod-cloud.net/