Discovering Data Sources in a Distributed Network of Heritage Information

Miel Vander Sande[1], Sjors de Valk [2], Enno Meijers[2], Ruben Taelman[1], Herbert Van de Sompel[3], Ruben Verborgh[1]

1: IDLab, Ghent University – imec, Belgium
2: national library of the netherlands
3: Data Archiving and Networked Services, The Netherlands

Abstract. The *Netwerk Digitaal Erfgoed* is a Dutch partnership that focuses on improving the visibility, usability and sustainability of digital collections in the cultural heritage sector. The vision is to improve the usability of the data by surmounting the borders between the separate collections of the cultural heritage institutions. Key concepts in this vision are the alignment of the data by using shared descriptions (e.g. thesauri), and the publication of the data as Linked Open Data. This demo paper describes a Proof of Concept to test this vision. It uses a register, where only *summaries* of datasets are stored, instead of all the data. Based on these summaries, a portal can query the register to find what data sources might be of interest, and then query the data directly from the relevant data sources.

1. Introduction

The Netwerk Digitaal Erfgoed (NDE) project rethinks the role cultural heritage institutions when exchanging data. Data from various institutions is no longer collected by a central aggregator and then shared with applications. Instead, the NDE targets a distributed setup where cultural heritage institutions are resposible for the publication of their data. In this demo paper, we present a Proof of Concept (PoC) that explores an architecture to supports this shift. We present a portal that wants to present information on fashion to end users and gathers all the relevant information on this theme from all the available datasets in the *network of cultural heritage*. For example, the portal can present information on body stockings, such as the fabric or type of clothing.

The cultural heritage institutions publish their datasets as Linked Open Data. They use *terms* from the NDE Network of Terms, which is developed by the NDE as the set of shared definitions that are relevant to cultural heritage data, such as for places, people, concepts and time periods. These terms have URIs, for example https://vtmk.data.momu.be/id/106061, which can be used to type an entity as a body stocking. Using this term URI, the portal can query the network to gather all the available data on this clothing type. For efficiency reasons in case many datasets are available, the portal first needs a list of datasets that might have interesting information. Then it can query these datasets to retrieve the information.

In the next section, we present an overview of our PoC's architecture. Next, in Section 3, we discuss how *dataset summaries* are used to find relevant datasets for a query and determine the importance of datasets. Then, in Section 4 we explain such datasets can be registered and how the portal obtains a list of relevant data sources. Finally, we conclude in Section 5.

2. Architecture

Our PoC architecture enables *Source Holders* (e.g. person, organization) that own or manage digital collections to publish *Datasets* in the cultural heritage network, and *Portal* clients to query data from all these *Datasets*. Portals query multiple, distributed heritage data collections via SPARQL. Thus, every data source should be able to handle queries and clients should be able to select the relevant data sources *before* query execution to avoid contacting irrelevant sources.

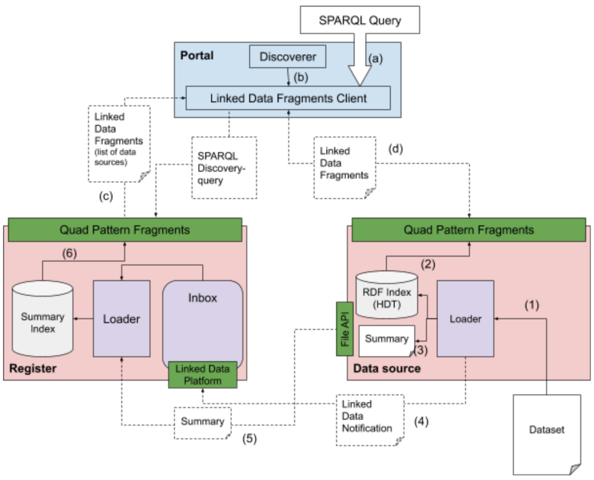

Fig. 1: General PoC architecture with all components and their interaction.

To achieve our requirements, we introduce three *distributed* components: a Web API where a *Dataset* can be found and queried as Linked Data (*Data Source*); an application that selects relevant data from the network and presents these to users (*Portal*); and a service that offers the selection of *Data Sources* that are relevant to a query, based on a registration of all the available *Datasets* and metadata of these *Datasets* (*Register*). This demo uses Triple Pattern Fragments (TPF) APIs [1] to expose *Data Sources*, which can be queried by a *Portal* with SPARQL by using a Linked Data Fragments client such as Comunica [2]. To enable the *Register* to make an informed decision on the relevant *Datasets* for a query, we explore a *Dataset* summary approach, where the *Register* retrieves a summary from each *Data Source*. Fig. 1 illustrates this general architecture and the two main interaction scenarios we will demonstrate: report a *Dataset* to the *Register*, and query data within the network of all *Data*

Sources.

Report a *Dataset* to the *Register*. *Source Holders* report *Datasets* to the network when they are new or when they want to disseminate an update. The process for both cases is equivalent. (1) A *Source Holder* loads a (new) version of the *Dataset* with the *Loader* component of the *Data Source*; (2) The *Loader* indexes these *Datasets* in the HDT format [3]: a binary, compact and searchable archive format for RDF data, which are published with a TPF interface; (3) The *Loader* creates a *Summary* of the *Dataset*, which is made available for download through a *File API*; (4) The *Data Source* sends a Linked Data Notification [4] to the *Inbox* of the *Register* via its Linked Data Platform API [5] with the message that a new *Dataset* or version is available; (5) The *Register* downloads the new *Dataset Summary* and adds it to the *Summary Index*; (6) The *Summary Index* is published using TPF.

Query data within the network of all *Data Sources*. *Portals* query the network to obtain the data they need. Hence, they first need to know which *Data Sources* are relevant to the query. A query is thus executed as follows: (a) A *Portal* sends a SPARQL query. (b) Based on this query, the *Discoverer* component of the *Portal* composes a discovery SPARQL query to select relevant *Data Sources*. For instance, this discovery query can contain the term URI about which the *Portal* wants to collect information. The triple pattern `<term URI> dcterms:isPartOf ?source` selects a list of *Data Sources* in which the term occurs. (c) With the Linked Data Fragments client and the discovery query, the *Portal* retrieves a list of relevant *Data Sources* from the *Register*. (d)The Linked Data Fragments client of the *Portal* then executes the original query on the selected list of *Data Sources* and returns the results to the *Portal*.

3. Dataset Summaries

This demo uses *Capability-based Dataset Summaries* as they are defined in the HiBISCuS system [6]. For each predicate we include the authorities of both subject and object URIs, which is denoted as a *capability*. The authority of a URI is its domain and optionally the port number and authentication information. The schema of the URI is added as a prefix, e.g. `http://`. An exception is the `rdf:type` predicate, where we include the entire object URIs. *Capability-based Dataset Summaries* only include superficial characteristics of RDF, i.e. only URIs and triples and no semantics or graph characteristics, and only require cheap operations like substring manipulation and string comparison. These summaries support checking whether *Datasets* have triples with a certain URI authority as subject and/or object; triples with a certain predicate and a certain URI authority as the subject and/or object; or have data of a certain type, as described by the predicate *capability*.

Unfortunately, this is insufficient in NDE, because many common types cannot be found through the `rdf:type` predicate, e.g. queries on periods of time or on the material type of an object. Furthermore, the predicates and URIs in the *Datasets* are quite homogeneous, so they do not sufficiently distinguish. As this demo focuses on *Data Sources* that use term URIs from the NDE Network of Terms, all sources have similar resources and URIs. Thus, we add a histogram of the object URIs (not the literals) of all triples, which represent term URI frequency, or, if needed, all triples that start with a certain prefix, e.g. of a thesaurus. Because histograms can become too large, we use a CountMinSketch [7], i.e. a compact, binary representation of a *Dataset* that allow determining how frequent an element (i.e. a term URI) is present. This compactness comes with a price in the form of false positives, meaning that the number of term URIs in a Dataset may be overestimated. False negatives do not occur, thus the absence of a certain URI in a *Dataset* is always certain, which is important to eliminate irrelevant *Data Sources*. In order to add a CountMinSketch to the *Summary*, they are encoded into a string with base64 encoding.

4. Registering and Obtaining Relevant Data Sources

In order to allow new *Datasets* to be added to the network, the *Data Sources* can send messages to the *Register* through Linked Data Notifications (LDN) [4]. They can notify the *Register* that a new version of a *Summary* is available, including its *Data Source* and where it can be found. The *Register* can download the new *Summary* and replace the old one in the *Index*. LDN uses the ActivityStreams vocabulary *(https://www.w3.org/ns/activitystreams)* for notifications of actions. The main notifications for our purposes are `as:Add` and `as:Update`. Listing 1 shows an example of an LDN from the Rijksmuseum Amsterdam to the *Register*.

```
@prefix as: <https://www.w3.org/ns/activitystreams#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

[] a as:Add; as:actor [ a as:Organization;
    as:name "Rijksmuseum Amsterdam"^^xsd:string ];
  as:object <link to summary> ;
  as:summary "Rijksmuseum Amsterdam added a summary"^^xsd:string .
```

Listing 1: Example of an LDN from the Rijksmuseum Amsterdam to the Register.

Portals can send requests to the *Register* to select relevant *Data Sources* with the SPARQL query language. To handle these requests, the *Register* publishes a TPF API to expose the *Summaries* of all the available *Data Sources*, by internally linking term URIs with *Datasets*. This enables *Portals* to request various information about the *Data Sources*, such as discovering all *Data Sources* that use a specific term URI.

For example, a *Portal* is interested to gather all the available information on body stockings. The *Datasets* that have clothing type information use the term URI https://vtmk.data.momu.be/id/106061 from the NDE Network of Terms, where the term URIs can be retrieved through a search API. The resulting interaction between a *Portal* and the *Register* is as follows: (1) the *Portal* sends a request to the *Register* for the triple pattern `<https://vtmk.data.momu.be/id/106061> dcterms:isPartOf ?source`; (2) the *Register* tests the CountMinSketch of all the *Data Sources* for the URI https://vtmk.data.momu.be/id/106061; (3) the *Register* adds all the matched *Data Sources* to the result, e.g. with the triple

`<https://vtmk.data.momu.be/id/106061> dcterms:isPartOf <http://demo.netwerkdigitaalerfgoed.nl/ldf/modemuze_momu>`; and (4) The *Register* sends the result to the *Portal*.

5. Conclusions

This demo illustrates the shift from a setup with a central aggregator towards a distributed setup for a *distributed network of heritage information*. It introduces a first possible solution to assist term-based queries. *Portals* can discover relevant *Data Sources* based on a term URI. These term URIs are agreed upon by the participating *Data Sources* and are available in the NDE Network of Terms. Term URIs are applied by the *Datasets* to type certain entities, such as fabrics of clothing or modes of transportation, and therefore occur in the object term of an RDF triple. Hence, Dataset Summaries include a CountMinSketch with object URIs. The *Register* can mark a *Data Source* as relevant by checking (a) the presence of a term URI and (b) possibly by how frequently it is used within the *Dataset*. In future developments, the Netwerk Digitaal Erfgoed will evolve the architecture to distribute the *Register* over the different *Data Sources* in the network, making a central authority obsolete. This of course includes more fine tuning of the *Dataset Summaries* and the source selection algorithms.

References

1. Verborgh, R., Vander Sande, M., Hartig, O., Van Herwegen, J., De Vocht, L., De Meester, B., Haesendonck, G., Colpaert, P.: Triple Pattern Fragments: a Low-cost Knowledge Graph Interface for the Web. Journal of Web Semantics. 37–38, (2016).
2. Taelman, R., Van Herwegen, J., Vander Sande, M., Verborgh, R.: Comunica: a Modular SPARQL Query Engine for the Web. In: Proceedings of the 17th International Semantic Web Conference (2018).
3. Fernández, J.D., Martínez-Prieto Miguel A, Gutiérrez, C., Polleres, A., Arias, M.: Binary RDF representation for publication and exchange (HDT). Web Semantics: Science, Services and Agents on the World Wide Web. 19, 22–41 (2013).
4. Capadisli, S., Guy, A., Lange, C., Auer, S., Sambra, A., Berners-Lee, T.: Linked data notifications: a resource-centric communication protocol. In: European Semantic Web Conference. pp. 537–553. Springer (2017).
5. Speicher, S., Arwe, J., Malhotra, A.: Linked Data Platform 1.0. W3C, https://www.w3.org/TR/2015/REC-ldp-20150226/ (2015).
6. Saleem, M., Ngomo, A.-C.N.: Hibiscus: Hypergraph-based source selection for sparql endpoint federation. In: European semantic web conference. pp. 176–191. Springer (2014).
7. Cormode, G., Muthukrishnan, S.: An improved data stream summary: the count-min sketch and its applications. Journal of Algorithms. 55, 58–75 (2005).

Vol-2451
urn:nbn:de:0074-2451-7

SEMPDS 2019
Posters and Demos at SEMANTiCS 2019

Proceedings of the Posters and Demo Track of the 15th International Conference on Semantic Systems
co-located with 15th International Conference on Semantic Systems (SEMANTiCS 2019)

Karlsruhe, Germany, September 9th to 12th, 2019.

Edited by

Mehwish Alam *,**,
Ricardo Usbeck ***,
Tassilo Pellegrini ****
Harald Sack *,**
York Sure-Vetter **

* FIZ Karlsruhe, Leibniz Institute for Information Infrastructure, Germany
** Karlsruhe Institute of Technology (KIT), Germany
*** Fraunhofer IAIS, Dresden, Germany
**** University of Applied Sciences St. Poelten, Austria
Both editors contributed equally

Table of Contents

Tabea Tietz, Jörg Waitelonis, Kanran Zhou, Paul Felgentreff, Nils Meyer, Andreas Weber, Harald Sack

- Discovering Data Sources in a Distributed Network of Heritage Information
  Miel Vander Sande, Sjors de Valk, Enno Meijers, Ruben Taelman, Herbert Van de Sompel, Ruben Verborgh
- How to Prepare an API for Programming in Natural Language
  Sebastian Weigelt, Mathias Landhäußer, Martin Blersch
- Generating ``Who Wants to Be a Millionaire?'' Questions Sets Automatically from Wikidata
  Markus Wohlan, Yannik Schröder, Frank Höppner
- Triple Pattern Join Cardinality Estimations over HDT with Enhanced Metadata
  Elena Wössner, Chang Qin, Javier Fernández, Maribel Acosta
- Visual Query Environment over RDF Data
  Kārlis Čerāns, Julija Ovcinnikova, Lelde Lace, Jūlija Hodakovska, Aiga Romane, Mikus Grasmanis, Elina Kalnina, Arturs Sprogis, Agris Sostaks

We offer a BibTeX file for citing papers of this workshop from LaTeX.

2019-09-05: submitted by Mehwish Alam, metadata incl. bibliographic data published under Creative Commons CC0
2019-09-22: published on CEUR-WS.org |valid HTML5|