Discovering data sources in a distributed network of heritage information

Proof of Concept, Dutch Digital Heritage Network

Miel Vander Sande (IDLab/Ghent University-imec)

Abstract

The Dutch Digital Heritage Network (Netwerk Digitaal Erfgoed - NDE) 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.

In the NDE vision each institution is responsible for the publication of its own data. The ultimate ambition is a setup without a central store where all digital collections are replicated. Instead, there will be a distributed network of information. Portals that want to provide end user services, can query the entire ‘network of heritage information’ to find the information that is relevant to their purposes.

This report describes a setup for 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 which data sources might be of interest and then query and retrieve the data directly from the data source.

The Proof of Concept was commissioned by NDE and has been developed by Miel Vander Sande (IDLab/Ghent University-imec), in collaboration with Ruben Verborgh (IDLab/Ghent University-imec), Herbert Van de Sompel (Data Archiving and Networked Services - DANS), Enno Meijers (NDE) and Sjors de Valk (NDE).
Introduction

The Dutch Digital Heritage Network rethinks the way cultural heritage institutions exchange data and thereby redefines their role. It moves away from the central aggregator setup, where the data from various institutions is collected in one place and then shared with applications. Instead, the NDE aims for a distributed setup where cultural heritage institutions get full control of and responsibility for the publication of their data.

A Proof of Concept (PoC) will explore the outlines of an architecture that supports this shift.

To illustrate the working of the setup, we use the example of a portal that wants to present information on fashion to end users. This portal wants to gather all the relevant information on this theme from all the available datasets in the ‘network of cultural heritage’. The portal might be interested in typing information, like fabric or type of clothing (e.g. body stockings).

The cultural heritage institutions publish their datasets as Linked Open Data. They use “terms” from the Network of Terms, which is developed by the NDE as the set of shared definitions that are relevant to cultural heritage data (i.e. for places, people, concepts, time periods etc). 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 type. For efficiency reasons, the portal first needs a list of datasets that might have interesting information. Then it can query these datasets to retrieve the information.
Components and their interaction

In the Proof of Concept (PoC), we enable Portal applications to query multiple, distributed heritage data collections via SPARQL. We distinguish several concepts in the network:

- **a Source Holder**: an organisation or person who owns and/or manages a digital collection, which they publish in the network;
- **a Dataset**: a digital representation of a collection that can be read, exchanged and processed;
- **a Data Source**: a server application which provides a location and service on the web where a Dataset can be found and queried;
- **a Portal**: an application that selects relevant data from the network to process them and to present the result to users;
- **a Register**: 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.

Two requirements are put forward to support these query capabilities:

1. Every Data Source should be able to handle queries on its Dataset and return a query-based selection of the data (Linked Data published by a Web API).
2. A Portal should be able to select the relevant Data Sources for its query before execution, in order to avoid querying irrelevant Data Sources (source discovery & selection).

The PoC enables the first requirement through Quad Pattern Fragments APIs on the Data Sources, that can be queried with SPARQL by using a Linked Data Fragments client. It enables the second requirement through a Register with a source selection service on top of Dataset summaries.

The interaction between the components in brief:

- **a Data Source**, managed by a **Source Holder** (e.g. a cultural heritage institution), publishes its own **Dataset** as Linked Data through a Web API (i.e. Triple Pattern Fragments).
- **Portals** query these **Data Sources** to gather information they need for running their application.
- Before querying the **Data Sources**, the **Portal** finds the **Data Sources** that are relevant to its query by querying the **Register**, which provides a source selection service.

The source selection service compels the Register to make an informed decision based on metadata from the Datasets. To enable this, we explore a **Dataset summary** approach, where the Register retrieves a summary from each Data Source.
Figure 1 describes the general architecture with the technical components. Two interaction scenarios are key, which consist of the following actions (the numbers in the list refer to Figure 1):

**(A) Report a dataset**
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 (Fernandez et al.): a binary, compact and searchable archive format for RDF data, which are published with a Triple/Quad Pattern Fragments interface.¹
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 to the Inbox of the Register via its Linked Data Platform API 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 as Triple/Quad Pattern Fragments.

**(B) Query the network**
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.

¹ [http://linkeddatafragments.org](http://linkeddatafragments.org)
Figure 1. General architecture.
Dataset Summaries

A Dataset Summary is a compact representation of an RDF Dataset, consisting of only its most important structure elements, characteristics or data. This information enables the Register to select the relevant Data Sources for a Portal query, without having to search all Data Sources completely or having to exchange all the data. The RDF datasets of the different sources can contain links to each other. However, each Data Source operates completely independent: they are not necessarily aware of others. This makes creating Summaries challenging, as a single source has no view on the data of others. Thus, it cannot determine what are links or what information is unique.

Figure 2 depicts two Data Sources which are linked, because they have certain URIs in common. Both have RDF triples where these URIs are used as objects. Each Data Source can decide which triples are useful to include in its Summary, because it can estimate which triples contain a link, based on only its own data.

![Figure 2. Datasets with common resources. These are used as objects in triples, so these are links.](image)

Dataset Summaries must also contain general information on the vocabulary (i.e., the used classes and properties), the URIs of the source data (e.g., the resources that are present and how they are identified), and the shape of the RDF data (i.e., what vocabulary information was applied to describe resources and the structure of the resulting graph). The challenge is to decide which information is useful to include in a Summary, because this depends on the needs and intentions of the Portals, i.e. the queries they want to execute. Depending on these needs, the required functionality of the Summaries may be different. Because the exact needs of future Portals are unknown, the Summaries must strike a balance between several aspects:
1. **Useful**: does the Summary facilitate the queries of the Portal(s)?
2. **Compact**: can the Summary be exchanged and processed in an efficient way that is fast enough?
3. **Computable**: can the Summary be generated in an easy way, that does not pose too much of a burden on the Data Source?

Of these three aspects, the first (usefulness) is essential and applies continuously over time. The second and third aspects (compactness and computability) apply only when a new Dataset becomes available or an existing Dataset has a newer version.

In this PoC we use basic Summaries as they are defined in the HiBISCuS (Saleem et al.) system. 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.

This is an example of information in a Summary:

```
@prefix ds: <http://semweb.mmlab.be/ns/summaries#>.

_:n3-0 ds:capability _:n3-1.
_:n3-1 ds:Predicate <http://id.loc.gov/vocabulary/relators/spn>;
  ds:sbjAuthority http://lod.kb.nl.
_:n3-0 ds:capability _:n3-2.
_:n3-2 ds:Predicate <http://purl.org/dc/elements/1.1/creator>;
  ds:sbjAuthority http://lod.kb.nl.
_:n3-0 ds:capability _:n3-3.
_:n3-3 ds:Predicate <http://purl.org/dc/elements/1.1/type>;
  ds:sbjAuthority http://lod.kb.nl;
_:n3-0 ds:capability _:n3-4.
_:n3-4 ds:Predicate <http://purl.org/dc/terms/contributor>;
  ds:sbjAuthority https://vtmk.data.momu.be;
_:n3-0 ds:capability _:n3-14.
_:n3-14 ds:Predicate <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>;
  ds:sbjAuthority http://lod.kb.nl, <http://thesaurus.europeanafashion.eu>,
  <https://vtmk.data.momu.be>;
  ds:objAuthority <http://www.europeana.eu/schemas/edm/Agent>,
  <http://www.europeana.eu/schemas/edm/Place>.
```

These basic Summaries are easy to compute and compact, because they only include superficial characteristics of RDF, i.e. only URIs and triples and no semantics or graph characteristics. They require only cheap operations like substring manipulation and string comparison.

The downside is that these basic Summaries support only a limited range of queries:
- Does the Dataset have triples with a certain URI authority as subject and/or object?
- Does the Dataset have triples with a certain predicate and a certain URI authority as the subject and/or object?
- Does the Dataset have data of a certain type, as described by the predicate rdf:type?

Unfortunately, this is insufficient for Portals in the cultural heritage network, because:
1. Many common queries on type cannot be found through the rdf:type predicate, e.g. queries on periods of time or on the material type of an object;
2. The predicates and URIs in the Datasets are quite homogeneous, so they do not sufficiently distinguish between Datasets.

This Proof of Concept will explore additions to the Dataset Summaries, to enable selection of Data Sources that might be relevant for several types of queries.

**Additions to Dataset Summaries**

To summarize: the basic Dataset Summaries are very restrictive in the type of queries they allow. They do not support queries on:
- Useful information in other predicates than rdf:type;
- URIs that are similar in authority, e.g. because they use the same vocabulary or URI source.

Summaries with additional information might:
- Decide dynamically based on relevance, for which predicates object and/or subject URIs should be included entirely, and not just the authority;
- Detect patterns in URI paths;
- Identify predictors on which triples have links to other Datasets.
- Adapt their included information to what is requested by the Portals.

One option is to add modules to the Summaries to support these functionalities.

From literature we know of several techniques that might be useful for the additions to the Summaries.
- **Resource Ranking, In & Out degree & TopK**
  The graph structure of the RDF Dataset may be used to estimate how important resources are. Several methods may be useful: general measures (e.g. directed graphs, incoming and outgoing links); general algorithms (e.g. Dijkstra or PageRank); or specific RDF statistics (Jun et al.; De Virgilio et al.; (Ermilov et al.); (Ben Ellefi et al.); (Hayes and Gutierrez).

- **URI Dissimilarity & Approximate string matching**
  In theory URIs are transparent, but in practice they often contain information (either textual or structural). The parts of an URI often reflect their origins, like their source or vocabulary. This means that textual differences may indicate links to other Datasets or may indicate other resources that can be found (Tsur).

- **Ontology distance & similarity**
  Data Sources may be distinguished by the similarities or dissimilarities between vocabularies (David and Euzenat). This means they can be used to determine which data are used where, and help selecting the relevant Data Sources for certain queries. The downside is that this requires a formal description of all vocabularies in the RDF (which is often not available). It also requires that the vocabularies are not too uniform and it requires collective access to the vocabularies of all the Data Sources.

- **Equivalence classes, entity comparison & similarity**
  Vocabularies may be used implicitly by searching for resources with common characteristics (Endris et al.); (Petrova et al.; Wang); (Gunaratna et al.). Within a Dataset similarities or dissimilarities between resources may be an indication of relevance or associations between resources.

In this PoC we focus on Data Sources that use Term URIs from the NDE Network of Terms. This means that the relevant resources and URIs are similar. The Summaries need to establish which Term URIs are in the Dataset (to eliminate irrelevant Data Sources) and how frequently they are used (to know the relevance of the Data Source). So the first addition to the Dataset Summary is a compact histogram of the Term URIs.
Occurrence and ranking of Terms in Datasets

Based on a Dataset Summary, the Register must be able to estimate:

- Whether a certain Term URI is used in the Dataset;
- (Optionally) the relevance of the Term URI in this Dataset compared to other Datasets.

This information can be deduced from a histogram that reflects the triple frequency of each object URI, because in triples the object often expresses a type by using a Term URI. However, histograms can become very large and we need to guarantee the compactness and speed of the Summary. So we use Approximate Membership data structures, which are compact, binary representations of a Dataset that allow us to determine whether an element (e.g. a Term URI) is present in the Dataset. The compactness we gain in this way comes with a price, because false positives may occur, meaning that the number of Term URIs in a Dataset may be overestimated. False negatives cannot happen, so it can always be determined with certainty that a certain URI is not present in a Dataset (which is important to eliminate irrelevant Data Sources).

Approximate Membership data structures can take several forms. We discuss the most relevant, in particular those that store both the presence of elements and their frequency.

- **Bloom filters & Counting Bloom filters**
  
  The original Approximate Membership data structure is the Bloom filter (Bloom). Per element in a set, different positions in a fixed length bit sequence are set to 1. These positions are calculated by several hash functions, which map an element to a specific number.

  Figure 3 shows an example of a sequence with length 10, using three hash functions per element. In its initial state, the bloom filter exists 10 bits long, all set to 0. When an element of the set is inserted, three different hash functions calculate the numbers 1, 2 and 9. Therefore, the bits at positions 1, 2 and 9 are set to 1. To check whether this element is part of the set or not, these positions are recalculated using the hash functions. When all positions are set to 1, the element is...
Because the output of hash functions can collide: different inputs can map to the same number, false positives can occur. The exact probability of the occurrence of false positives depends on a combination of the number of elements, the length of the bit sequence and the number of hash functions. A Bloom filter only allows to register whether elements are present or not. A Counting Bloom filter (Bonomi et al.) is an extension, which also stores the frequency of each element and allows for deletion of elements. Counting Bloom filters are less compact, however.

- **CountMinSketch**
  The CountMinSketch (Cormode and Muthukrishnan) uses the same techniques as a BloomFilter, but applies these to a histogram instead of a set. This allows for determining the individual cardinality of each element. CountMinSketch is still fairly compact.

In this PoC we use CountMinSketch because it offers all of the required functionality. For each Data Source we generate a CountMinSketch with the object URIs (not the literals) of all triples. Or, if needed, all triples that start with a certain prefix, e.g. of a thesaurus.
These are hashed to a string with base64 encoding, then they are supplemented with metadata and added to the Summary. The example below is a CountMinSketch where the probability of false positives is 5%, using the MurMur hash function (Appleby).

```prefix ms: <http://semweb.mmlab.be/ns/membership>
_:membershipFilter a ms:CountMinSketch; # AMF metadata
ms:hashSize 524288;
ms:hashFunction <MyMurmur1>, <MyMurmur2>;
ms:memberCollection [  
  ms:sourceCollection _:n3-0; # connect to summary  
  ms:projectedProperty rdf:object;  
  ms:projectedTermtype rdfs:Resource  
];
ms:falsePositiveRate 0.05;
ms:falseNegativeRate 0.0;
ms:binaryRepresentation "QmF...ZTY"^^xsd:base64Binary.
```
Notifications

The Data Sources can send messages to the Register through Linked Data Notifications (LDN)\(^3\). 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 for notifications of actions\(^4\). The main notifications for our purposes are as:Add and as:Update.

The example below shows 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 .
```

---

\(^3\) [https://www.w3.org/TR/ldn/](https://www.w3.org/TR/ldn/)

\(^4\) [https://www.w3.org/ns/activitystreams](https://www.w3.org/ns/activitystreams)
The Register interface

Portals can send requests to the Register to select relevant Data Sources, as shown in Figure 1. These requests use the SPARQL query language. (Please note that Portals use SPARQL both for requests to the Register and to the Data Sources).

To handle these requests, the Register publishes a TPF API (Triple Pattern Fragments) for the Summaries of all the available Data Sources. This enables Portals to request various information about the Data Sources.

As an example: a Portal is interested to gather all the available information on body stockings. The Datasets that have type-information use Term URIs from thesauri or controlled vocabularies, e.g. the Term URI https://vtmk.data.momu.be/id/106061 for body stockings. As mentioned before, in this PoC we focus on Term URIs that are in the NDE Network of Terms, where the Term URIs can be retrieved through a search API.

The Register registers the Datasets with these Term URIs and offers look-up functionality. So a Portal can discover all the Data Sources that use a specific Term URI.

```
SELECT * {
}
```

This query from a Portal to the Register consists of several steps:

- The Portal sends a request to the Register for the Triple Pattern Fragment <https://vtmk.data.momu.be/id/106061> dcterms:isPartOf ?source
- The Register tests the CountMinSketch of all the Data Sources for the URI https://vtmk.data.momu.be/id/106061
- The Register adds all the positive 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> .
- The Register sends the result to the Portal.

Currently, the PoC is limited to simple queries. Hence, it does not support derivatives of certain terms (narrower or broader) showing up in the query results. Extensions such as inference based on domain ontologies are possible and will be considered in follow-up stages.
Conclusion

This Proof of Concept explores an architecture that realizes the shift from a setup with a central aggregator towards a distributed setup. The NDE describes this vision as a "distributed network of heritage information". Key aspects are

- institutions get full control and responsibility over the access to their data;
- the aggregator no longer replicates data, but evolves to an intelligent entrypoint and cache for the data sources in the network.

In brief, the Proof of Concept uses several components. A Data Source, managed by a Source Holder (e.g., a cultural heritage institution), publishes its own Dataset as Linked Data through a Web API (i.e. Triple Pattern Fragments). Portals will query these Data Sources to gather the information they need for running their application. To facilitate this, the Register provides a source selection service to Portals to inform them about the Data Sources that are relevant to their query. Therefore, the Register needs to make an informed decision based on metadata from the Datasets, for which we explore a Summary approach.

The Register retrieves a Dataset Summary from each Data Source. A Dataset Summary is a representation of a dataset that should be compact (i.e. small enough for efficient exchange), simple to compute (i.e. the overhead to the Data Source is limited) and applicable (i.e. the information in the Summary enables the Register to determine if a Data Source is relevant). However, finding a good trade-off between these three aspects is challenging. When Datasets are homogeneous in vocabulary and URIs, it is harder to tell them apart with a minimum of information. For instance, the list of URI authorities per predicate between Data Sources is too identical. Hence, the applicability is low.

We discussed two complimentary mitigation strategies: (a) use more advanced statistics and profiling based on vocabulary-level, graph structure or string/entity (dis)similarity, and (b) use a modular approach where Summary information is added based on the queries a Data Source receives.

This Proof of Concept 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 should include a compact representation of the 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.

Therefore, we encode this information in a Approximate Membership Filter (i.e. Bloom Filter or CountMinSketch), which is a binary representation of a set to test the presence of an element with a certain false positive probability. These filters are very compact, efficient to
create and probe. However, they can cause the Register to return some irrelevant Data Sources (false positives). In practice, though, the resulting overhead will be limited to one irrelevant Data Source in some cases, which can be dismissed early on with a more clever query algorithm.

In future developments, the Dutch Digital Heritage Network 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.
Works Cited


Hayes, Jonathan, and Claudio Gutierrez. “Bipartite Graphs as Intermediate Model for RDF.”


