Discoverability of SPARQL Endpoints in Linked Open Data

Heiko Paulheim[1] and Sven Hertling[2]

[1] University of Mannheim, Germany
Research Group Data and Web Science
heiko@informatik.uni-mannheim.de
[2] Technische Universität Darmstadt
Knowledge Engineering Group
hertling@ke.tu-darmstadt.de

Abstract. Accessing Linked Open Data sources with query languages such as SPARQL provides more flexible possibilities than access based on derefencerable URIs only. However, discovering a SPARQL endpoint on the fly, given a URI, is not trivial. This paper provides a quantitative analysis on the automatic discoverability of SPARQL endpoints using different mechanisms.

1 Introduction

Query languages such as SPARQL provide efficient ways of accessing Linked Open Data sources. In his *design issues* document from 2006, Tim Berners-lee states that "to make the data be effectively linked, someone who only has the URI of something must be able to find their way the SPARQL endpoint." [2].

However, automatically discovering the SPARQL endpoint for a given resource is still not a trivial problem. Approaches to solve that problem include:

- Standardized vocabularies for describing datasets, such as *VoID*, where the descriptions are provided at URLs that can be canonically derived from a URI [1], and
- Catalogs of datasets such as *datahub*[1] or the LATC data source inventory[2], which can be queried for a SPARQL endpoint with a given URI.

In this paper, we explore the success rates of those strategies using a large representative sample of URIs in Linked Open Data, and discuss the results. Furthermore, we propose a simple, multi-strategy resolution service which delivers SPARQL endpoints for URIs.

2 Strategies for Discovering SPARQL Endpoints

We examine two basic strategies for discovering SPARQL endpoints from a URI: trying to retrieve VoID descriptions, and leveraging external catalogs of datasets.

[1] http://datahub.io/
[2] http://dsi.lod-cloud.net/

2.1 Retrieving VoID Descriptions

The VoID specification [1] recommends to use the RFC 5785 standard [5] for publishing discoverable VoID descriptions of a dataset. This means that a URI of the form `http://hostname/.well-known/void` is to be used for publishing VoID descriptions. Although the specification states that the `/.well-known/void` path segment should be located at the root level, i.e., directly follow the host name part of the URI, our experiments have shown that it is sometimes located at deeper locations. Thus, we use the following approach for trying to discover VoID vocabularies:

> Given a URI, remove the portion after the last slash (`/`), and append `.well-known/void`. If no VoID description is found at that location, and there are segments left after the host name, continue from the start.

For example, given the URI `http://www.example.org/data/xyz`, we would try the following URLs for retrieving a VoID description, using the VoID [1] and Provenance [3] vocabularies:

1. `http://www.example.org/data/.well-known/void` and
2. `http://www.example.org/.well-known/void`,

assuming that the first URL does not return a VoID description.

As a second strategy to retrieving VoID descriptions, we retrieve the RDF dataset from the (dereferencable) sample URI, and look for one of the following axioms:

1. `?x void:inDataset ?d`
2. `?x prv:containedBy ?d`[3]

Although, in the literature, means other than VoID descriptions have been proposed to link data to SPARQL endpoints [4], we do not expect them to be too widely spread, since they are not backed by a standardization document.

2.2 Leveraging External Catalogs

Catalogs of datasets, such as *datahub*, list datasets as well as their metadata, including SPARQL endpoints, if applicable. For our prototype, we use the *datahub catalog*, which lists data sets as well as their SPARQL endpoints. Similar searches could be issued on any catalogs of Linked Open Data.

We have implemented all those strategies in the *SEnF* (SPARQL Endpoint Finder) service, a simple web service which can be used to retrieve SPARQL endpoints for a URI.[4]

[3] We do not demand that `?x` is connected to `<URI>`, e.g., by a `rdfs:definedBy` statement, in order to make this approach as versatile as possible, and since we assume that a VoID description linked from a dataset will in most cases be the description of that dataset, and not of another one.

[4] `http://tinyurl.com/sparqlsenf`

Table 1. Results on different strategies for finding SPARQL endpoints on 10,000 random URIs, reporting both the number of URIs for which *any* SPARQL endpoint was found, as well as the number of URIs for which a *valid* SPARQL endpoint was found. The numbers in parantheses denote the total number of endpoints found.

Strategy	Datahub Catalog	`/.well-known/void` (all)	`/.well-known/void` (standard)	Link to VoID
# found	7,389 (26,124)	110 (392)	94 (288)	9 (9)
# valid	1,375 (2,978)	53 (106)	53 (72)	0 (0)

3 Quantitative Analysis

We have tested the approaches discussed above on a random sample of 10,000 subjects in the 2012 billion triple challenge dataset,[5] which we deem a representative sample of Linked Open Data in the wild. Out of those 10,000 URIs, 8,893 were dereferencable.

For each endpoint retrieved by any strategy, we have checked the correctness of the result by issuing a query of the form `ASK {<URI> ?r ?x}` at the endpoint, and consider the returned endpoint as a valid result if `TRUE` is returned upon the query.

Table 1 shows the results of our evaluation. The first observation is that in many cases and by most strategies, more than one endpoint is returned, which shows that there is some redundancy in terms of SPARQL endpoints (i.e., more than one endpoint may contain information on a resource).

The main observation is that using external catalogs clearly outperforms other methods in terms of coverage, being able to locate endpoints for 74% of all URIs. However, only in 14% of the cases, at least one of the retrieved endpoints[6] was online during our experiment[7] and actually contains data about the resource in question, which also demonstrates the limitations of the approach.

The approaches using VoID and the provenance vocabulary are still not adopted on a large scale, thus, the coverage of those approaches is much lower. On the other hand, the data found by following `/.well-known/void` is much more precise than those delivered by catalogs, showing a precision of 0.48 (in contrast to 0.19 for the catalog based approach). The approach looking for direct links to VoID descriptions provided information on endpoints in some cases, however, the SPARQL statement for checking the validity of the endpoint failed in those nine cases because the original URI was redirected, and the redirect URI, which pointed to the dataset, not the resource, was not found in the endpoint.

Furthermore, we can observe that there is a deviation between the standard specification for providing VoID descriptions (i.e., providing them at the server's root directory), and the actual deployment (in some cases, they are located at deeper levels). This may hint at a practical problem with implementing the

[5] `http://km.aifb.kit.edu/projects/btc-2012/`

[6] In some cases, more than one endpoint is retrieved.

[7] Carried out between July 22nd and July 23rd, 2013

standard, i.e., hosting data sets on servers for which the authority providing the data set does not have root access rights.

It is further remarkable that for no URI in our sample, an endpoint could be retrieved by every strategy. This shows that there is a need to use multiple strategies in parallel, like our implementation of the *SEnF* service does.

4 Conclusion

The capability of locating SPARQL endpoints for a given URI has been stated as a desired property of Linked Open Data. In this paper, we have evaluated several strategies for performing that *URI-to-endpoint* resolution, based on a large random sample of the Billion Triple Challenge Dataset.

Approaches using proposed methods such as VoID and the provenance vocabulary are scarcely in use (and sometimes not implemented according to the specification), they lead to a valid SPARQL endpoint in less than 1% of all cases. That finding means that catalogs are essential for discovering SPARQL endpoint, at least in the short and medium term. However, although performing better than the approaches mentioned before, catalogs also do not provide information in sufficient quality at the time being.

Overall, we were not able to locate suitable SPARQL endpoints in most of the cases – for more than 85% of all URIs, no SPARQL endpoint could be found. The reasons may be two-fold: (i) it is not possible to discover the endpoints with the methods described in this paper, or (ii) no such endpoints exist. While in many cases, the latter case is likely (e.g. for single FOAF documents at websites, or blogging software that publishes RDF(a), but does not provide a SPARQL endpoint), it is beyond the scope of this paper (if not completely infeasible due to the open world assumption) to make a statement about the actual availability of SPARQL endpoints for Linked Open Data URIs.

Our evaluation has furthermore shown that no single strategy outperforms all other strategies. Thus, for practical purposes, using multi-strategy approaches such as the SEnF service is the most suitable way for discovering endpoints. Since the SEnF service follows a modular architecture, new catalogs and/or resolution strategies may be plugged in as they become available and/or standardized.

References

1. Keith Alexander, Richard Cyganiak, Michael Hausenblas, and Jun Zhao. Describing Linked Datasets with the VoID Vocabulary. `http://www.w3.org/TR/void/`.
2. Tim Berners-Lee. Linked Data. `http://www.w3.org/DesignIssues/LinkedData.html`.
3. Olaf Hartig and Jun Zhao. Provenance Vocabulary Core Ontology Specification. `http://trdf.sourceforge.net/provenance/ns.html`.
4. Kjetil Kjernsmo. The necessity of hypermedia RDF and an approach to achieve it. In *Proceedings of the First Linked APIs Workshop*, 2012.
5. Mark Nottingham and Eran Hammer-Lahav. RFC 5785 – Defining Well-Known Uniform Resource Identifiers (URIs). `http://tools.ietf.org/html/rfc5785`.