A decentralized infrastructure for the efficient management of resources in the web of data

Michalis Stefanidakis
Ionian University, Dept. of Computer Science
Plateia Eleytherias, Corfu, 49100, Greece
mistral@ionio.gr

Ioannis Papadakis
Ionian University, Dept. of Archives and Library Science
Ioannou Theotoki 72, Corfu, 49100, Greece
papadakis@ionio.gr

ABSTRACT

A key requirement in linking open data refers to the rendering of such data easy to explore and consume. Such a requirement is critical for the overall quality of linked data in terms of consistency and interlinking. Although the provision of dereferenceable URIs is a step towards the right direction, current status of the semantic web indicates that such a feature alone is not enough. After all, as Berners Lee stressed, a URI is just an identifier for a resource not a recipe for its retrieval.

In this paper, a decentralized infrastructure is presented capable of efficiently managing the URIs that are being consumed within the web of data. The proposed work is based on the *registry*, a place where triplestores may store information about their local URI entities. Such information evolves around *backlinks* that are defined as the references that are made from triples of remote triplestores to the locally defined URIs of a given triplestore. The registries communicate with each other through a lightweight protocol that facilitates the management and retrieval of URI entities across the LOD cloud.

Categories and Subject Descriptors

H.3.4 [**Systems and Software**]: Distributed Systems

General Terms

Design

Keywords

registry, protocol, linked data, backlinks

1. INTRODUCTION

During the past few years, the attention of the semantic web community seems to be focused on the concept of linked open data – LOD. Researchers envision a web of data – WoD where both users and agents will be able to issue semantic queries in order to retrieve useful information. In this direction, a number of data providers have created and accordingly exposed publicly available, federated datasets (i.e. triplestores) featuring querying capabilities and being able to exchange information with each other in an interoperable fashion through the employment of well established protocols and standards like URIs, RDF and SPARQL. According to the latest statistics[1], currently exist more than 311 datasets that contain more than 32 billion triples.

The realization of the vision that governs the generation of linked data requires proper and meaningful interlinking between the URIs that are introduced within each dataset. Without proper interlinking, LOD datasets will remain loosely coupled islands within the vast sea of the WoD. Efficiently integrating information that exists within more than one dataset will inevitably be a privilege of very few, centralized, highly equipped, third-party services capable of efficiently replicating and accordingly manipulating data from multiple datasets.

The compulsory employment of dereferenceable URIs for the identification of the various resources that constitute the cell of the information existing within each LOD dataset, certainly facilitates interoperability between diverse datasets. However, the employment of URIs alone is not enough. The lack of adequate protocols and tools that would automate the process of locating and consequently reusing 'useful' URIs, has led to the current status of linked data, where the participating datasets are very poorly integrated with each other. Even if some of these datasets are interlinked, this is often the result of a lot of hard and time-consuming manual work [4].

In this paper, an interlinking protocol is presented, capable of underpinning a decentralized infrastructure for the WoD. The proposed work evolves around the *registry*, a place where triplestores may store information about their local URI entities. The registries communicate with each other through a lightweight protocol that facilitates the management and retrieval of URI entities across linked data repositories.

The rest of this paper is structured as follows: the next section identifies the problem of loosely coupled triplestores across the LOD cloud. Then, a number of approaches are outlined that are targeted towards the confrontation of such an issue. Section 4 presents a decentralized infrastructure for managing URIs within linked data, by giving special attention to its integral components, namely the *registry* and

[1]LOD cloud statistics: `http://lod-cloud.net/state`, accessed at: 2.2.2012

SWIM 2012, May 20, 2012, Scottsdale, Arizona, USA.

the *registry communication protocol*. The following section demonstrates the viability of the proposed approach through the presentation of the BTC 2011 case study. Section 6 addresses the security and trust issues that should be taken under consideration and section 7 discusses various issues that arize from the introduction of the proposed approach. Finally, the last section summarizes the paper and points directions for future work.

2. DISCOVERING USEFUL, REMOTE URIS

The true power of linked data is their inherent ability to overcome the barriers of their origin repositories[2] and participate in aggregative structures that ultimately provide value added services to their users. However, before one can aggregate triples in the context of a particular semantic service, one has to locate the right ones in the underlying LOD cloud. In an ideal situation, repositories would be highly interlinked, meaning that they would define their own URIs (i.e. local URIs) only if such URIs would not exist in triples of the remaining LOD cloud. Consequently, they would employ the elsewhere-defined URIs in their own triples (i.e. remote URIs), or, associate them with a locally-defined URI entity through an owl:sameas (or equivalent) predicate.

Despite the existence of various off-the-shelf tools (e.g. triplestore engines, SPARQL endpoints) that render LOD data immediately available to potential stakeholders, the aforementioned situation is far from reality. It seems that existing repositories do not make extensive use of remote URIs, and, even if they do, they do not do it right (for example, see [10]).

It is the authors' belief that there is a lack of adequate protocols and tools that would facilitate the creation of LOD services capable of discovering 'useful' remote URIs across diverse repositories resulting this way in the creation of highly interlinked repositories. It is argued that such services should not be based on the centralized model. Centralized approaches towards the enhancement of interlinking across LOD repositories exhibit certain drawbacks deriving from their centralized nature that renders them in the long term non-scalable and expensive to maintain.

In order to address the interlinking problem in a scalable fashion, a decentralized approach is proposed, that is based on a decentralized registry infrastructure. The registry is defined as a place where repositories may store information about their local URI entities in an open and expandable way. The registry of each repository is accessible to remote stakeholders through an accordingly defined protocol that will be presented later in this paper.

3. CURRENT STATUS IN INTERLINKING TRIPLESTORES

The most naive approach in locating and accordingly aggregating triples that belong to different triplestores would be to issue a SPARQL query to the local endpoint of a given URI in order to retrieve one or more literals that are somehow related to this URI. Then, use these literals as input to a second SPARQL query that would be addressed to other, potentially relevant SPARQL endpoints, in order to find remote URIs that are likely to be related to the initial one. Although this approach is relatively easy to implement since it only depends on the availability of public SPARQL endpoints, it suffers from certain drawbacks. First of all, one has to know in advance all the triplestores and their corresponding SPARQL endpoints that contain possibly useful URIs. Despite the recent emergence of the Vocabulary of Interlinked Datasets – voiD [14] tools (e.g. [2]) it is still too early to expect a voiD description from every repository. Moreover, this approach not only is time consuming but also requires a significant amount of manual intervention, in order to overcome disambiguation problems regarding the retrieved URIs [3].

3.1 The Centralized Approach

Other approaches follow the centralized model in order to offer services capable of aiding users and agents in interlinking LOD repositories. For example, the OKKAM entity name system – ENS [4] proposes a service capable of creating upon request and maintaining unique URIs for resources mentioned within LOD repositories. Such identifiers are stored in a distributed infrastructure of synchronized nodes. Thus, according to this scenario, all LOD providers rely on the ENS service in order to receive unique identifiers for their resources. Consequently, in an ENS-powered world, searching for possibly useful URIs within the LOD cloud would be a matter of a few queries addressed to the ENS service. The challenging part about this service is how to persuade the LOD providers to rely on a third-party entity to manage their URIs.

Another, quite popular service is sameas.org[3], which accepts a given URI and accordingly provides a list of equivalent URIs existing in remote triplestores. Such a service crawls triplestores, stores and indexes URIs that participate in triples with owl:sameas as their predicate. Like any other centralized solution, this service suffers from scalability issues concerning the synchronization of its contents with the actual contents of the LOD cloud.

Apart from sameas.org, a number of other approaches offer LOD interlinking services that are based on information deriving from the aggregation of information of the repositories of the LOD cloud (Sindice [8], subj3ct.com[4], kasabi.com[5]). Such services try to automatically or semi-automatically discover interlinking factors across diverse repositories that, in a perfect LOD world, should have been native to each repository since the time of their creation.

3.2 The Silk Framework

The Silk [9] link discovery framework supports LOD providers in creating RDF links between entities of their own repository and entities that belong to remote ones. Silk accesses triplestores via the SPARQL protocol and can thus be employed against local as well as remote SPARQL endpoints.

The Silk framework promotes a promising decentralized option in enhancing the interlinking across repositories of the LOD cloud. Perhaps the effectiveness of the approach would be greater if it could aid LOD providers in locating the

[2]The term repository is used to describe an autonomously managed set of semantic data and services. Such data goes beyond triples stored in triplestores, as the repository is envisioned to have additional storage components and also data in forms other than triples.

[3]sameas online service: `www.sameas.org`, accessed at: 2.2.2012

[4]subj3ct.com: `http://www.subj3ct.com`, accessed at: 2.2.2012

[5]kasabi.com: `http://kasabi.com/`, accessed at: 2.2.2012

remote triplestores that contain possibly useful information.

3.3 LOD Backlinks

Quite recently, a number of approaches emerged trying to increase the degree of interlinking between LOD repositories through the identification and management of the backlinks of their URIs ([1], [5], UK PSI backlinking service[6], Dipper[7]). In the LOD context, backlinks are defined as the references that are made from triples of remote triplestores to the locally defined URIs of a given triplestore. The importance of backlinks has been initially identified by Google through the implementation of its famous PageRank algorithm for ranking search results [6]. The social media community also realized that backlinks could significantly improve the connectivity of its underlying information resources. Thus, various social media vendors utilized pingback or traceback mechanisms that enable authors of a weblog entry or article to obtain immediate feedback, when other people reference their work [5].

The authors of [5] mimic the social media pingback mechanism and suggest that whenever a reference to a remote resource is made from a local triplestore, the corresponding remote triplestore should be notified and update its underlying repository with accordingly produced triples. Such an approach faces trust issues, since remote triplestores should be granted write access to the triplestore they reference.

There are also two online services, namely the UK Public Sector Information – UK PSI backlinking service and Dipper that provide backlinking information. Both of them harvest and accordingly analyze a limited number of triplestores in order to extract the desired backlinking information. Consequently, both of the approaches are only tested in a smaller scale and it remains unknown how well they would scale in the entire LOD cloud.

It is the authors' belief that backlinks will greatly enhance connectivity among triplestores. However, the highly dynamic and diverse nature of the LOD cloud dictates that any effort that aims in underpinning the interlinking between triplestores should be applied at an infrastructural level in order to remain scalable and effective for the wider LOD community.

4. THE PROPOSED APPROACH

The work presented here is based on a decentralized registry infrastructure capable of enhancing the interlinking between LOD repositories. The proposed infrastructure manages the backlinks that are attributed to a local entity URI from triples existing in remote repositories via an accordingly designed protocol. It is argued that the proposed approach could be employed from various LOD-powered services, wishing to exploit bidirectional interlinking between repositories. In addition to backlinking management, the registry infrastructure may host a textual index service, mapping from free-text queries to the relevant URIs. The latter service is an essential part of the enhanced interlinking vision, by connecting human free-text input to the precisely defined entity URIs.

The proposed registry functionality is layered as shown in

[6]enakting: `http://backlinks.psi.enakting.org/`, accessed at: 2.2.2012

[7]dipper: `http://api.talis.com/stores/iand-dev1/items/dipper.html`, accessed at: 2.2.2012

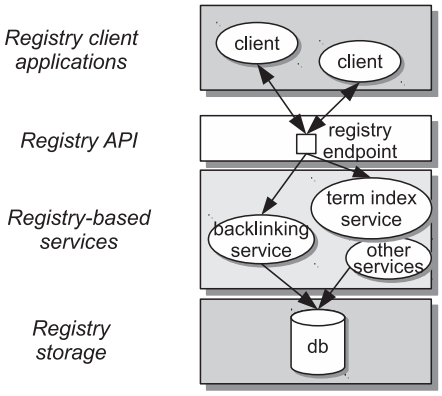


Figure 1: The layered stack of Registry components

(Figure 1) and includes the following components:

- The *Registry storage*, holding the local URI index together with interlinking information, be it backlink counts or a label term index.
- A set of *Registry-based services*, which can be deployed in a modular fashion and according to the actual needs of each LOD repository. The endpoints of these services complement the typical SPARQL endpoint and can be described via voiD.
- A *Registry communication protocol*, a protocol used to exchange data with the registry services endpoint.
- *Registry client applications*, making use of but also updating the registry information in a truly read/write mode.

For performance reasons, the registry storage component is instantiated as a conventional relational database and will not be described further. The following sections provide a detailed description of the functionality of the cornerstone backlinking and term index services, as well as the specification of the protocol used to underpin the Registry API.

4.1 Registry Backlinking Support

The triplestore of each LOD repository holds triples that consist of its own entity URIs (i.e. *local URIs*) and URIs that are introduced in remote repositories (i.e. *remote URIs*). The proposed approach dictates the existence of a Registry for every LOD repository. Each entry of the Registry contains a local URI and its corresponding backlinks. Such information includes the SPARQL endpoint of the remote repository that made the backlink (i.e. referenced the specific local URI) and the number of references to the particular local URI by the same remote repository. This way, a repository knows the number of backlinks to its local URIs and the way to access such backlinks. Figure 2 presents a condensed view of an imaginary registry-enabled LOD cloud containing three repositories. According to the proposed infrastructure, each repository consists of a triplestore and a Registry.

More specifically, repository C defines the local URI *C:r*, which is accordingly stored in repository C's Registry. The triplestores of repository A and B contain triples that reference *C:r*. Thus, repository C's Registry should store the

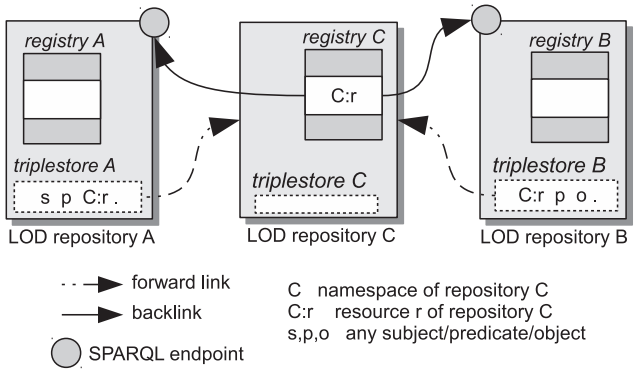


Figure 2: Interlinking LOD repositories via the Registry-based infrastructure

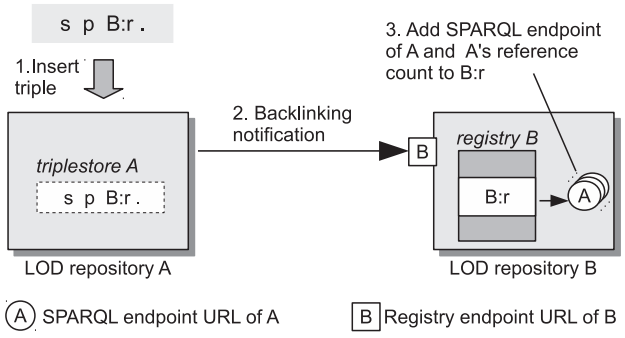


Figure 3: The backlink notification mechanism

SPARQL endpoints of repositories A and B. The LOD cloud in its current status contains repositories that are capable of providing information about the remote URIs of a certain triplestore (i.e. *forward links* in Figure 2). The Registry-based infrastructure enables any repository of the LOD cloud to provide the backlinks of its local URIs (see Figure 2).

The proposed infrastructure also introduces a notification mechanism that is responsible for notifying the appropriate Registry in the event of a backlink creation. When a LOD repository creates a triple containing a reference to a remote URI, the notification mechanism sends a message to the remote URI's Registry containing a) the URI being referenced, b) the SPARQL endpoint of the repository that made the reference and c) the total count of the repository's references to this URI. Upon reception, and given that the repository's policy accepts the message, the local Registry stores the SPARQL endpoint of the remote repository that referenced the URI, together with the number of its references (Figure 3).

The information exchange between a LOD repository and a client application for querying or updating backlinking information is accomplished through an accordingly designed communication protocol that is presented in section 4.3.

4.2 Registry Assisted Text-to-URI Mapping

In the current state of the LOD cloud and despite the continuous interlinking efforts, there exists a major hurdle in the usage of LOD by end-user applications. Human user input contains mostly ambiguous free-text terms, while the LOD cloud interlinking scheme is based on crisp URIs. Consequently, the interlinking status of the LOD cloud cannot be maximized without taking into consideration the fact that the entrypoint of many end-user LOD applications requires raw, error-prone text input.

A text-to-URI mapping service can bridge the gap between end-users and repositories. The same service can also be beneficial to machine agents; consider the case of automated entity alignment between different LOD repositories.

In the proposed infrastructure, a term index hosted by the Registry associates a searchable text structure with the local URIs, providing thus the aforementioned text-to-URI translation. The indexing terms are extracted from the labels of entities – the term *label* signifies here not only the text connected to the traditional *rdfs:label* predicate but virtually any kind of meaningful textual attribute.

What distinguishes the text-to-URI mapping service from other LOD services is the ambiguous nature of the corresponding results. Excluding the rare case of exact textual matches without synonyms, the results normally contain a set of possible replies, each one coupled with a ranking value – a situation similar to common web searches. The information exchange protocol described in the next section returns the full result set back to the requesting application.

Table 1: Generic Parameters for any Registry Service Request

`type`	(mandatory) defines the direction of data flow: **query** for reading from and **update** for writing to the Registry
`component`	(mandatory) defines the targeted Registry service. Each enabled service has a unique identifying name
`registry`	(optional) selects a specific repository in case the Registry accommodates more than one repositories

4.3 The Registry Communication Protocol

The communication protocol between a client application (agent) and a Registry service endpoint defines a simple yet extensible API that encompasses all possible data exchanges with any Registry service in read /write mode. In all cases, a client submits an HTTP GET request of the generic form:

```
http://ex.org/registry?type=[query type]&
  component=[service name]&[other query-specific params]
```

The generic parameters of the HTTP GET request are shown in Table 1.

The request may have more parameters, depending on the query type. The communication protocol does not specify these additional parameters, allowing for a modular expansion of the Registry with new services. Examples for these parameters will be given in the following sections, where the protocol transaction for the backlinking and term index services will be shown.

The HTTP response carries the requested data (in the case of a read request) or a success code (for a write request). In the following examples, for clarity reasons, the response is XML-formatted, while the protocol allows for alternative forms, depending on content negotiation.

4.3.1 Requests for a Backlinking Service

For the support of the fundamental backlinking service, the Registry protocol allows a client to query the number of backlinks per remote LOD repository for a particular URI. The parameter *uri* (skipping for clarity actual URL encoding) is appended to the generic parameters of the request:

```
http://ex.org/registry?type=query&
  component=backlinks&uri=http://ex.org/res/myresource
```

The reply returns the number of backlinks of the selected URI per remote LOD repository:

```
<?xml version='1.0'?>
<response status='ok' uri='http://ex.org/res/myresource'>
  <answer rrepository='http://remoterepository1'>
   10</answer>
  <answer rrepository='http://remoterepository2'>
   90</answer>
  ...
  <answer rrepository='http://remoterepositoryN'>
   150</answer>
</response>
```

The Registry protocol also supports notifications to a Registry in order to update the number of backlinks of a local URI that come from a remote LOD repository. In this case, the request is formed as follows:

```
http://ex.org/registry?type=update&
  component=backlinks&uri=http://ex.org/res/myresource&
  rrepository=http://remoterepository&count=100
```

In the above request, the *uri* parameter specifies the URI being referenced by the remote LOD repository defined by the parameter *rrepository*, while the parameter *count* specifies the number of backlinks to this URI existing in the remote repository.

The reply contains a success of failure message, depending on write authorization to the Registry data, e.g:

```
<?xml version='1.0'?>
<response status='ok' />
```

4.3.2 Request for a Text-to-URI Mapping Service

A text-to-URI mapping service allows a client application to query the registry about specific URIs matching with a free-text parameter:

```
http://ex.org/registry?type=query&
  component=label&text=Paris&
```

In this case, the reply may be ambiguous, therefore the returned results are ranked by a weighted value:

```
<xml version='1.0'>
<response status='ok' text='Paris'>
  <answer weight='1' label='Paris'>
http://ex.org/res/uri1</answer>
  <answer weight='1' label='Paris'>
http://ex.org/res/uri2</answer>
  <answer weight='0.9' label='Paris'>
http://ex.org/res/uri3</answer>
  <answer weight='0.7' label='Parisienne'>
http://ex.org/res/uriN</answer>
</response>
```

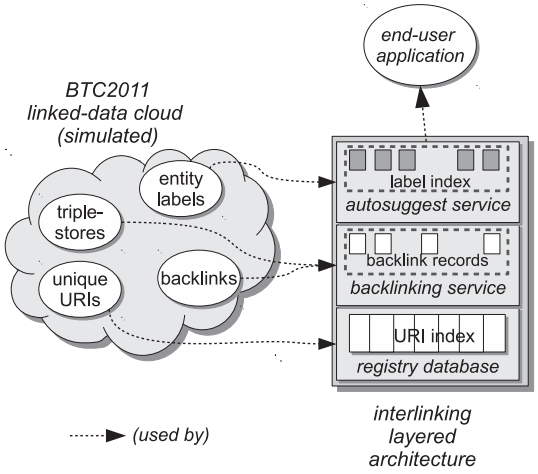


Figure 4: The major components of the Registry-based application.

5. THE BTC 2011 CASE STUDY

In order to demonstrate the feasibility of the proposed approach, a Registry-based application has been created on top of the Billion Triple Challenge - BTC 2011 Dataset[8]. The goal of the application is to aid end-users in discovering meaningful URIs within the BTC 2011 dataset. The phrase 'meaningful URIs' is interpreted as URIs that have an explicit meaning to human users. In order to identify such URIs within the entire dataset, it was decided to filter out URIs that appear as subjects in triples where objects are some kind of literal and the corresponding predicate refers to some kind of label (e.g. *rdfs:label*).

The application is based on the proposed layered architecture and makes use of the accordingly created Registry of each repository within the BTC 2011 dataset. More specifically, the Registry's main purpose is to enable a distributed backlinking service which in turn underpins an autosuggest service and end-user application that aid users in retrieving the most popular URIs within the BTC dataset. Popularity is measured according to the number of backlinks of each local URI. The architecture consists of two major components (Figure 4):

- A *simulated linked-data cloud*, based on the entire BTC 2011 dataset. This cloud provides the data needed for the underlying services and the corresponding end-user application.
- A set of *Registry-based services* and an end-user application demonstrating the concept of the proposed approach.

In the subsequent sections, the architecture, functionality and justification of the aforementioned components are presented.

5.1 The Simulated LOD cloud

The term *simulated LOD cloud* describes a dataset consisting of URIs, their corresponding labels and how many

[8]Billion Triple Challenge 2011 Dataset: `http://km.aifb.kit.edu/projects/btc-2011/`

times they appeared in the triplestores (i.e. 'provenance' triplestores) that constitute the dataset. This dataset has been created by processing the entire BTC dump and acts as the data source of the actual components of the layered architecture of the Registry-based application. More details about the way the simulated LOD cloud has been processed can be found in [11]. The process revealed 102,773,693 unique URIs, from which 87,888,452 belong to BTC 2011 provenance triplestores.

5.1.1 Labels of URIs

Human-readable labels are a significant aid for the usage of linked data, be it in the case of an end-user application or of a triplestore maintenance tool [13]. In the proposed end-user application, the main focus is to prove the significance of a URI look-up "autosuggest" (or "autocomplete") service for humans based on the corresponding URI labels.

Such services heavily depend on textual labels describing URIs. Despite the fact that most labels exist within triples where the corresponding predicate semantically adheres to the concept of a label (i.e. rdfs:label or semantically equivalent predicates), nevertheless, a significant number of such predicates belong to other, domain-specific namespaces [11]. Thus, the process of label extraction within the BTC 2011 dataset heuristically searched and accordingly identified 10,931,614 unique URIs with labels. Such number is considered adequate to demonstrate the "ranking by popularity" service that is based on the proposed infrastructure.

5.2 Registry-based Services

The BTC 2011 application builds upon the decentralized Registry infrastructure that is described throughout this paper. At the bottom level of the corresponding architecture finds its place the Registry, a component containing all the URIs that are owned by a triplestore. This information is exposed to a set of services wishing to promote interlinking among LOD repositories, namely a backlinking service and an autosuggest service serving the corresponding end-user application. The implementation details of the above services are presented in the following sections.

5.2.1 The Registry

The *Registry* keeps an index of all the local URIs of a triplestore together with information concerning the backlinks within the BTC 2011 dataset. For the needs of the BTC 2011 application, the Registry is realized as a mysql database powered by the sphinx indexer[9].

5.2.2 The Backlinking Service

The *backlinking service* answers requests for backlinks of specific local URIs within a triplestore. It utilizes the Registry by attaching backlinking information to local URIs. When a backlink of a URI is discovered, the Registry is updated with information concerning the backlink's provenance repository and the number of times the URI has been referenced in triples of the remote triplestore. The backlinking service's endpoint is implemented as a service, which queries the Registry about the backlinking information of a local URI and accordingly receives the corresponding result set in order to pass it to the autosuggest service. The twisted framework[10] is employed for the issuing of the necessary queries to the Registry and the consequent communication with the autosuggest service.

5.2.3 The Autosuggest Service

The *autosuggest service* provides human-searchable access to the URIs of a triplestore. It is a service that should be implemented by triplestores of the LOD cloud, needing to provide the first hop from user freetext input to the world of linked data. URIs that are targeted towards human consumption (e.g. subject headings from the Library of Congress[11] look like this: `http://id.loc.gov/authorities/subjects/sh94002414`) are not necessarily formatted in a human-readable way like the one endorsed by DBpedia (a typical DBpedia URI looks like this: `http://dbpedia.org/resource/paris`). Users are not always able to discover a URI within the WoD in a straightforward fashion. Thus, it is argued that a text-to-URI mapping service is essential to the accessibility of the LOD cloud. Such services are already present in popular sites of the traditional web such as Google, Bing, Yahoo and Wikipedia.

The autosuggest service acts as a demonstrator that proves the feasibility of the overall decentralized infrastructure. According to the proposed service, the Registry is enhanced with a searchable text index, which is built from the labels of the underlying URIs. The service's endpoint addresses two queries; one is targeted to the Registry and asks for URIs that their label matches the user's input. The second query asks from the backlink service the backlinking information of each matched URI. Both the responses are accordingly merged and tunnelled to the autosuggest end-user application. Again, the twisted framework is employed for the issuing of the necessary queries to the Registry and the consequent communication with the backlinking service as well as the autosuggest end-user application.

5.2.4 The Autosuggest End-user Application

The autosuggest end-user application[12] utilizes the proposed architecture in order to provide its users with a way to retrieve URIs from the simulated LOD cloud ranked by their popularity. More specifically, the proposed application provides the opportunity to address queries consisting of some characters to the entire cloud or to a certain repository. The users are able to choose whether they want the resulting URIs to be ranked according to the number of their backlinks or, alphabetically ranked. The end-user application is an ajax-based web application, which exchanges information with the aforementioned autosuggest service (see Figure 5, Figure 6, Figure 7).

6. SECURITY AND TRUST ISSUES

When a client requests to write to the Registry, according to the protocol presented in earlier sections, authentication and authorization issues arise. A protection mechanism is typically needed, in order to allow only trusted clients to update the Registry of a LOD repository. The proposed Registry protocol does not explicitly incorporate security mechanisms; on the contrary, the authentication and autho-

[9]Sphinx indexer: `http://sphinxsearch.com`, accessed at: 2.2.2012

[10]twisted: `http://twistedmatrix.com`, accessed at: 2.2.2012

[11]LCSH: `http://id.loc.gov`, accessed at: 2.2.2012

[12]Autosuggest app: `http://thalassa.ionio.gr/ranked`

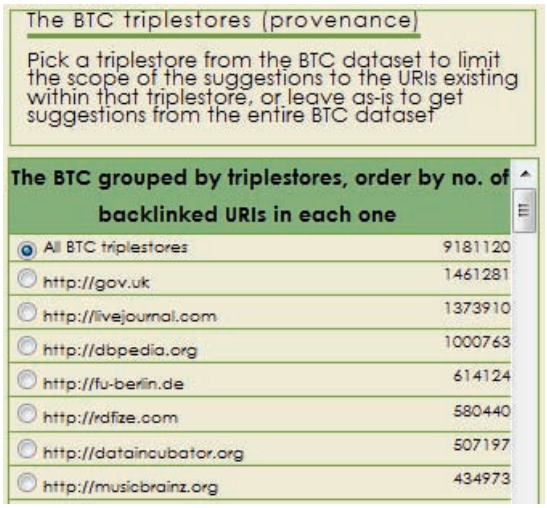


Figure 5: The BTC 2011 triplestores' selection pane

Figure 6: The BTC 2011 URI labels' autosuggest conrtol

rization functionality is assumed to be implemented in the underlying transfer framework.

This design decision frees the proposed approach from the burden of adopting a currently existing security mechanisms or implementing a custom one. The former solution confines the approach to a mechanism that could be outdated in the future, while the latter has the risk of being error-prone.

Currently, there are evolving standards and practices especially conceived to enable read-write access to LOD repositories: FOAF+SSL [12], although initially designed for the social semantic web, is suitable for client application authentication when connecting to the Registry services. At the same time, the WebAccessControl (WAC)[13] decentralized system and ontology can be deployed in a LOD repository to enable authorization access control. For backlinking Registry annotations, a well-known LOD dataset hub like the Data Hub[14] (the former CKAN) could be queried by a LOD repository implementing WAC, in order to accept update requests from sites with LOD datasets already registered in the hub. Of course, other trust-alliances could emerge in the future.

7. DISCUSSION

A major contribution of the proposed infrastructure is its' inherent ability to promote interlinking between LOD repositories through the systematic management of backlinks. Such ability is not limited to the obvious quantitative increase of the number of links between URIs existing in remote LOD repositories. For example, the employment of backlinks in the crawling process provides a way out of possible dead-ends caused by current, uni-directional interlinking. Moreover, semantic inconsistencies between different URIs that nevertheless refer to the same thing may be easier identified and accordingly resolved through the employment of backlinks. It is evident that the systematic handling of backlinks that is proposed in this work does not affect the autonomous nature of the resources participating in the WoD. After all, backlinks could be considered as yet another case of triples linking resources to the remote triplestores that consume them.

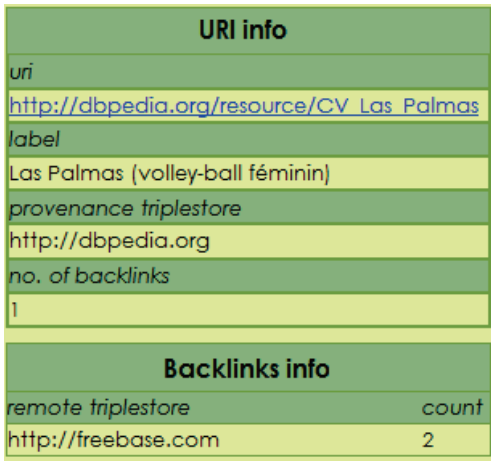
At this point it should be mentioned that the described work does not address the "quality" issue of the generated backlinks. However, such an issue could be resolved from the emergence of future services that would follow the principles of the proposed infrastructure. The modular nature of the proposed approach facilitates the emergence of other, value-added services that underpin the usefulness of the Registry. For example, the case study presented in this work could be enhanced to address SPARQL queries on top of suggestions.

As far as consumption of resources is concerned, the full-scale deployment of the proposed infrastructure is estimated to increase the network traffic of the corresponding service endpoints. Such increase is expected to be high for popular sites acting as "hubs" in the WoD. However, these sites should be able to efficiently deal with this matter, since the exponential popularity of linked data in the past few years made them capable of quickly adjusting to additional needs for resources. After all, LOD repositories with "popular" URIs are motivated to support the extra traffic, since they will benefit the most from the efficient management of backlinks. The case of dumping data in bulk from external datastores to LOD repositories could be confronted by services capable of batching backlink notifications to the corresponding remote LOD repositories, or, by exchanging backlink notifications offline. Moreover, the realization of the proposed case study shows that any given Registry of the LOD cloud could be efficiently managed by existing database systems, even for big LOD repositories like DBpedia.

Finally, it is clear that the proposed infrastructure copes with evident needs of the WoD. Major LOD repositories have already started adding similar functionality to their products (e.g. freetext indices or provenance tracking records). Yet, the benefits of decentralized, Registry-based services will reach their full potentials only in the case of a global-scale adoption. Fortunately, global endorsement of the proposed approach does not depend on its deployment by the great number of LOD repositories in the LOD cloud but rather, it depends on two major factors; the adoption of the proposed functionality by the relatively small number of major triplestore software products (e.g. sesame, virtuoso,

[13] `http://www.w3.org/wiki/WebAccessControl`, accessed at: 6.12.2011

[14] `http://thedatahub.org/`, accessed at: 6.12.2011



URI info	
uri	
http://dbpedia.org/resource/CV_Las_Palmas	
label	
Las Palmas (volley-ball féminin)	
provenance triplestore	
http://dbpedia.org	
no. of backlinks	
1	

Backlinks info	
remote triplestore	count
http://freebase.com	2

Figure 7: The URI's backlinking information table

4store), and the intention of LOD repositories to make use of the provided functionality.

8. CONCLUSIONS

In this paper, it is argued that the LOD community lacks the appropriate tools and protocols that would enable LOD repositories to be highly interlinked by (re)using elsewhere defined URI entities. Along these lines, a decentralized infrastructure is presented that utilizes a lightweight communication protocol. The proposed approach is capable of enhancing the interlinking between diverse LOD repositories. The protocol facilitates information exchange between the Registries of the LOD cloud. The Registry is defined as a place where LOD repositories may store information about their local URI entities in an open and expandable fashion. Along these lines, a set of protocol-based services based on the Billion Triples Challenge 2011 dataset were implemented.

Various security and trust issues that arise from the deployment of the proposed approach were mentioned and accordingly discussed. The proposed approach is meant to inspire the design of adequate services that are addressed both to LOD providers and to end-users. The decentralized nature of the proposed approach dictates that its success relies on the adoption from the wider LOD community, which in turn depends on the provision of the appropriate APIs and tools that would facilitate such an adoption[15].

9. REFERENCES

[1] M. Stefanidakis, I. Papadakis. Linking the (un)linked data through backlinks. In Proceedings of the International Conference on Web Intelligence, Mining and Semantics, WIMS '11, pages 61:1–61:5, New York, NY, USA, 2011. ACM.

[2] C. Boem, J. Lorey, F. Naumann. Creating voiD descriptions for Web-scale data. Web Semantics: Science, Services and Agents on the World Wide Web, Vol. 9, no. 3, September 2011, pp. 339-345, 2011, ISSN 1570-8268, doi: 10.1016/j.websem.2011.06.001, available at:http://www.sciencedirect.com/science/article/pii/S1570826811000370.

[3] Y. Raimond, C. Sutton, M. Sandler. Automatic Interlinking of Music Datasets on the Semantic Web. In: Proceedings of the 1st Workshop about Linked Data on the Web - LDOW2008, doi=10.1.1.123.9753, 2008.

[4] P. Bouquet, H. Stoermer and B. Bazzanella. An entity name system (ENS) for the semantic web. In Proceedings of the 5th European semantic web conference on The semantic web: research and applications (ESWC'08), Sean Bechhofer, Manfred Hauswirth, Joerg Hoffmann, and Manolis Koubarakis (Eds.). Springer-Verlag, Berlin, Heidelberg, pp. 258-272.

[5] S. Tramp, P. Frischmuth, T. Ermilov and Sauren Auer. Weaving a social data web with semantic Pingback. In Proceedings of the 17th international conference on Knowledge engineering and management by the masses (EKAW'10), Philipp Cimiano and H. Sofia Pinto (Eds.). Springer-Verlag, Berlin, Heidelberg, 2010, pp. 135-149.

[6] L. Page, S. Brin, R. Motwani and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web.Stanford InfoLab technical report, 1999, available at: http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf.

[7] D. Le Phuoc, A. Polleres, M. Hauswirth, G. Tummarello, C. Morbidoni. Rapid prototyping of semantic mash-ups through semantic web pipes. In: Quemada, J., LeÃşn, G., Maarek, Y.S., Nejdl, W. (eds.) WWW, pp. 581-590. ACM, New York (2009).

[8] G. Tummarello, R. Delbru and E. Oren. Sindice.com: weaving the open linked data Proceedings of the 6th International Semantic Web Conference (2007).

[9] J. Volz, C. Bizer, M. Gaedke and G. Kobilarov. Silk - A Link Discovery Framework for the Web of Data. In 2nd Workshop about Linked Data on the Web (LDOW2009), Madrid, Spain, 2009.

[10] H. Halpin, P. J. Hayes, J. P. McCusker, D. McGuinness and H. Thompson. When owl:sameAs Isn't the Same: An Analysis of Identity in Linked Data. In Proceedings of ISWC 2010, Lecture Notes in Computer Science, Vol. 6496/2010, pp. 305-320.

[11] M. Stefanidakis, I. Papadakis. An autosuggest service based on LOD backlinks. Contestant in Semantic Web Challenge - Billion Triples track, hosted in International Semantic Web Conference 2011, 23-27 Oct., Bonn, Germany, 2011, available at: http://iswc2011.semanticweb.org/fileadmin/iswc/Papers/PostersDemos/swc/swc2011_submission_14.

[12] H. Story, B. Harbulot, I. Jacobi and M. Jones. FOAF+TLS: RESTful Authentication for the Social Web. In Proceedings of the First Workshop on Trust and Privacy on the Social and Semantic Web, 2009.

[13] B. Ell, D. Vrandecic, and E. Simperl, Labels in the Web of Data, in Proceedingsof the 10th International Semantic Web Conference (ISWC2011), Lecture Notes inComputer Science, Berlin / Heidelberg, 2011, Springer.

[14] K. Alexander, R. Cyganiak, M. Hausenblas, and J. Zhao. Describing Linked Datasets: On the Design and Usage of voiD, the "Vocabulary Of Interlinked Datasets". In WWW Workshop: Linked Data on the Web, 2009.

[15]Registry-based tools are available through the Project's homepage, available at: `http://swrg.ionio.gr`