An autosuggest service based on lod backlinks

Ioannis Papadakis[1] and Michalis Stefanidakis[2]

[1] Ionian University, Dept. of Archives and Library Science
Ioannou Theotoki 72, Corfu, 49100, Greece
[2] Ionian University, Dept. of Computer Science
Plateia Eleytherias, Corfu, 49100, Greece
{papadakis,mistral}@ionio.gr

Abstract. Augmented and meaningful interlinking between the triple-stores of the lod cloud is vital to the success of the linked-data movement; the employment of URIs alone is not enough to integrate datasets and make them accessible to humans and machines. In this paper, a layered interlinking architecture is presented, based on the concept of a *registry*, a place where linked-data services can store information about entities in an open and expandable way.
As a demonstrator for the Billion Triple Challenge, an autosuggest application is presented (`http://thalassa.ionio.gr/ranked/`), capable of enhancing the interlinking among diverse datasets through the utilization of backlinks, i.e. references to the URIs of a local dataset originating from remote datasets.

Keywords: backlinks, linked-data, registry, autosuggest, autocomplete

1 Introduction

The compulsory employment of 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 reusing such 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 linked, this is often the result of a lot of hard and time-consuming manual work [2].

The vision of a highly interlinked lod cloud depends on the number of references from within a local dataset to URIs that are introduced in remote ones. It is therefore apparent that the necessary tools should be provided that would facilitate the automated discovery of "useful", remote URIs. Such tools would not only be employed from lod dataset providers, but also from end-user applications wishing to provide transparent access to the underlying lod cloud.

Some approaches follow the centralized model in order to offer services capable of aiding users and agents in interlinking lod datasets. For example, the OKKAM entity name system ENS [2] proposes a service capable of creating upon request and maintaining unique URIs for resources mentioned within

lod datasets. Another, quite popular service is sameas.org[3], which accepts a given URI and accordingly provides a list of equivalent URIs existing in remote datasets. 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.

Quite recently, a number of approaches emerged trying to increase the degree of interlinking between lod datasets through the identification and management of the backlinks of their URIs ([1], [3], UK PSI backlinking service[4], Dipper[5]). The importance of backlinks has been initially identified by Google through the implementation of its famous PageRank algorithm for ranking search results.

In this paper, an autosuggest end-user application is proposed capable of enhancing the interlinking among diverse datasets through the utilization of backlinks. Backlinks are defined as the references that are made to the URIs of a local dataset from remote ones [1]. End users of the application are able to type some letters of a word or phrase they have in mind, and instantly retrieve the corresponding suggestions from a registry of all native URIs, ranked by the number of each URI's backlinks. When compared to the suggestions a user would get in the case of retrieving just alphabetically ranked URIs, it is evident that the proposed approach succeeds in filtering a lot of useless information that exists in the entire BTC dataset. Moreover, as it will be described later in this paper, the realization of the autosuggest application for the BTC dataset is engineered in a modular way that promotes scalability should it be endorsed by the data providers of the online lod cloud.

2 The proposed layered interlinking architecture

In order to demonstrate the effectiveness of the proposed approach, a registry of native URIs has been created for every triplestore within the BTC dataset. The registries interact with a distributed backlink 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 native URI. The proposed architecture consists of two major components (Fig. 1):

- A *simulated linked-data cloud*, based on the entire BTC dataset [6]. This cloud provides the data needed for the underlying services and the corresponding end-user application.
- A *registry*, a set of *web 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.

[3] sameas online service: www.sameas.org, accessed at: 29.9.2011
[4] enakting: http://backlinks.psi.enakting.org/, accessed at: 29.9.2011
[5] dipper: http://api.talis.com/stores/iand-dev1/items/dipper.html, accessed at: 29.9.2011
[6] Billion Triple Challenge 2011 Dataset: http://km.aifb.kit.edu/projects/btc-2011/

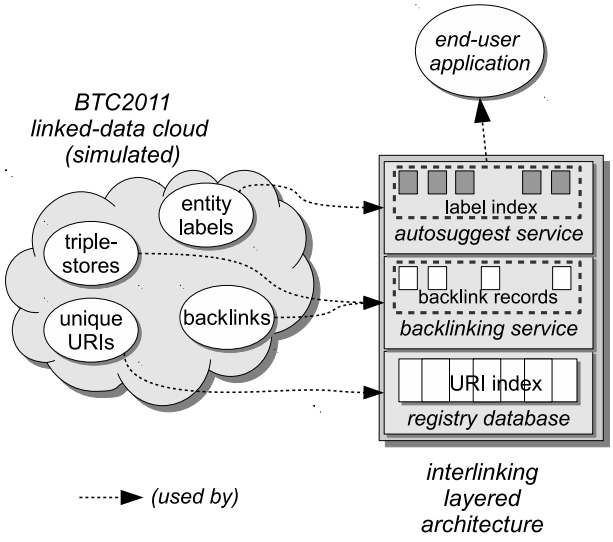

Fig. 1. The major components of the proposed infrastructure.

2.1 The simulated linked-data cloud

The term *simulated linked-data cloud* describes a dataset consisting mostly of entity URIs and their appearance counts in the 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 that is described throughout this paper.

Processing was performed as an initial off-line step and is decoupled from real-time web services and front-end application. This step has the sole purpose of building the simulated environment. As it will be argued in the following paragraphs, this method of environment building is unnecessary in a real-world setup, since online triplestores possess by definition all necessary dataset information.

The building process of the simulated linked-data cloud is split into the following steps:

- Identification of the *provenance triplestores* that appear as sources of BTC triples.
- Grouping of all*unique entity URIs*, according to their provenance triplestores.
- Identification of triples that contain *foreign URIs*, i.e. references to entity URIs not belonging to the provenance triplestore of the triple.
- Discovery of *URI labels*, enabling human user access to entity URIs.

Provenance Triplestores In the simulated environment created from the BTC dataset, provenance triplestores are perceived as autonomous organizations with

their own linked-data management policies, in an analogy to Internet's Domain Name System – DNS. Indeed, the provenance URI of each quad within the BTC dataset is processed in order to create the simulated set of provenance triplestores. Each provenance triplestore corresponds to a domain name that derives from the original provenance URI of a quad by keeping only the first two levels of the dot hierarchy of the domain name of the corresponding URI (e.g. `a.example.org` and `b.example.org` are considered manifestations of the same triplestore named collectivelly `example.org`).

Due to the aforementioned method for building provenance URIs, the resulting set of provenance triplestores bears an accurate resemblance to the real linked-data cloud on the web. The accuracy of the provenance URIs is essential, because the proposed registry services depend on detailed knowledge of triplestores and their interlinking.

Unique URIs In actual web setups, the association between a linked-data URI and its owner triplestore is conventionally identified by its' namespace. In a similar manner, the simulated linked-data cloud records all entity-owner triplestore associations. The process revealed 102,773,693 unique URI references, from which 87,888,452 belong to BTC provenance triplestores. The top 10 provenance triplestores, ranked by number of URIsthey own, account for the 86% of total unique URIs in the BTC dataset. Tracking the owner triplestores of entity URIs is needed in the simulated linked-data cloud since it enables the creation of a distributed URI index per owner triplestore.

Technically, in the context of linked-data every URI reference is an entity reference. Practically however, a web page URL, an ontology namespace URI and a normal linked-data entity URI may have different significance for a consuming application. In the simulated linked-data cloud an all-URI-encompassing approach was taken. This is one case where a real-life's linked-data repository could fare better, by knowing the type of any URI it holds; the fact urges for better data entry tools that can annotate URIs entered in the triplestore with a sense of meaning.

Foreign URIs In any triple contained in a tripestore, a *foreign* entity reference is a URI belonging to a different triplestore. In the simulated linked-data cloud, triples with foreign URIs are processed in order to extract *backlinking* information. A backlink typically informs a triplestore that one of its own entities is being referenced in another triplestore. This knowledge facilitates crawling and distributed querying between triplestores. In this publication, another usage of backlinking is demonstrated: URI ranking by backlink popularity.

For extracting backlinkig information, the whole BTC dataset has been parsed in order to locate triples with foreign URIs. When such a reference was detected, it was recorded together with the triplestore that made the reference (i.e. the provenance triplestore of the particular triple). The same process can be executed in batch mode on real-life triplestore dumps. However, it is evident that such a static scenario would affect scalability. A dynamic approach such as

the one proposed in [1], is better suited for the decentralized and ever-evolving nature of the online lod cloud.

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. In the proposed end-user application, the main focus is on a search tool with "autosuggest" (or "autocomplete") functionality. This kind of tools depends heavily on textual labels describing linked-data entities. For this reason, as many labels as possible were extracted from the BTC dataset by examining the predicates of each triple. The process of label extraction searched a heuristically compiled set of predicates and identified 10,931,614 unique labeled entities. The summary of predicates and label counts per predicate are presented in Table 1.

Table 1. Predicates and label counts per predicate

Predicate	Label Count
`http://open.vocab.org/terms/sortLabel`	58,519
`http://purl.org/dc/elements/1.1/title`	2,062,880
`http://www.fao.org/aims/aos/languagecode.owl#hasEnglishName`	7,642
`http://www.w3.org/1999/02/22-rdf-syntax-ns#label`	6
`http://www.w3.org/2000/01/rdf-schema#label`	8,540,192
`http://www.w3.org/2004/02/skos/core#altLabel`	1,517
`http://www.w3.org/2004/02/skos/core#prefLabel`	259,895
`http://zeitkunst.org/bibtex/0.1/bibtex.owl#title`	963

What is immediately apparent in Table 1 is the diversity of predicates used to attach textual information to linked-data entities. The list of predicates is by no means exhaustive and also contains some hard-to-guess predicates. Moreover, labels can be attached to internal blank nodes, making them impossible to use for characterizing an entity. This fact stresses the difficulty of bulk label extraction from a random triplestore without prior knowledge of its label coding conventions. In the online lod cloud a better approach would be to keep track of any text that can be used as a label at the time of data entry in a triplestore.

The provided BTC dataset suffered also from the fact that there is no guaranty that all labels of BTC entities found their way into the dataset. Although this is not the case with real-life triplestores and consequently scalability is not affected, nevertheless it was necessary for the needs of this challenge to use additional external linked-data dumps containing entities mentioned in BTC dataset, in order to enrich the label set with additional 1,201,381 labels.

2.2 Registry-based Web Services

The ultimate goal of the demonstrated setup is to show the effectiveness of the layered architecture, which is described throughout this paper. At the bottom level of the proposed architecture finds its place the registry, a component containing all entity URIs that are owned by a triplestore. The registry's database can store various types of "interesting" information, depending on application needs. This information is exposed to a set of middleware services wishing to promote interlinking among lod datasets. Such services are in turn employed by application-specific services that expose interlinking functionality to end-user applications and agents. The overall layered architecture is designed to be scalable with an open scheme of pluggable modules, which are outlined below:

The Registry The *Registry* keeps an index of all the native entity URIs of a triplestore together with information concerning the backlinks within the BTC dataset. The modular nature of the registry provides the opportunity to store other information about each native entity URI, besides its backlinks. Such information depends on the nature of the corresponding services. The registry is realized as a MySQL database powered by the Sphinx indexer[7].

The Backlinking Service The *backlinking service* answers requests for backlinks of specific local entity URIs within a triplestore. It utilizes the registry by attaching backlinking information to local entity URIs already existing in the registry. When a backlink of a URI is discovered, the registry is updated with information concerning the backlink's triplestore (i.e. provenance information) and the number of times the entity URI has been referenced in triples of the remote triplestore. The backlinking service's endpoint is implemented as a REST-style web service, which queries the registry about the backlinking information of a local entity URI and accordingly receives the corresponding result set in order to pass it to the autosuggest service. For the needs of the Billion Triples Challenge, the twisted framework[8] has been employed for the issuing of the necessary queries to the registry and the communication with the autosuggest service.

The Autosuggest Service The *autosuggest service* provides human-searchable access to the entity URIs of a triplestore. Moreover, it adds to the registry a searchable text index, built from the labels of the entities. The service's endpoint is implemented as a REST-style web service, which asks the registry about entity URIs that their label matches the user's input. Moreover, it requests from the backlink service the backlink information of each matched entity URI. Both the responses are accordingly merged and tunneled to the autosuggest end-user application. For the needs of the Billion Triples Challenge, the twisted framework has been employed for the issuing of the necessary queries to the registry

[7] Sphinx indexer: http://sphinxsearch.com, accessed at: 29.9.2011
[8] twisted: http://twistedmatrix.com, accessed at: 29.9.2011

and the consequent communication with the backlinking service as well as the autosuggest end-user application.

The Autosuggest end-user application The proposed autosuggest end-user application[9] utilizes the layered architecture which is described throughout this paper in order to provide its users with a way to retrieve URIs from the simulated linked data 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 triplestore. 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 according to a request/response protocol, that issues http GET requests and retrieves json and xml responses.

3 Conclusions

In this paper, an autosuggest application for the BTC dataset is presented, which ranks the URIs that exist within the triples of the BTC dataset according to the number of their backlinks. The proposed application enhances the interlinking between the various triplestores that constitute the BTC dataset by utilizing the underlying backlinks.

The main concern behind the implementation of the proposed application was the need to support it with a scalable architecture that could easily be adopted by the entire, online lod community. For this reason, one of the first design decisions that was made, dictated that the implementation should be based on infrastructural components applicable to every lod triplestore of the lod cloud. Along these lines, the selected approach is based on a modular architecture that can be easily adopted from existing lod dataset providers and capable of being extended with yet-to-come applications focusing on the enhancing of interlinking between lod triplestores.

References

1. Stefanidakis, M., Papadakis, I.: 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. Bouquet, P., Stoermer, H. and Bazzanella, B.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.

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

3. Tramp, S., Frischmuth, P., Ermilov, T. 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, 2010, pp. 135-149.

Appendix: Addressing the evaluation requirements

1. *The applications must make use of the Billion Triple Challenge 2011 Dataset.*
   The entire BTC dataset has been parsed and fed into an accordingly designed database. Autosuggest application: while the user types some letters of a word or phrase, he is presented URI suggestions, ranked by the number of each URI's backlinks.
2. *The tool or application has to make use of at least the first billion triples.*
   The autosuggest application maintains an alphabetically ranked index of 102M unique URIs together with their backlinking information, that were extracted from the entire BTC dataset.
3. *The tool or application is allowed to use other data.*
   For unlabeled URIs belonging to the BTC dataset, external sources (dbpedia, musicbrainz dumps) were used to acquire these labels.
4. *Theapplication does not have to be specifically an end-user application.*
   The proposed autosuggest application is a highly interactive, end-user application.

- *The application should do more than simply store large numbers of triples.*
  The proposed application manages the URIs that constitute the underlying triples and ranks them according to their backlinks
- *The application or tool(s) should be scalable.*
  The end-user application is based on a modular layered architecture that can be endorsed by all lod dataset providers. Besides the centralized approach that has been inevitably employed for the needs of the BTC dataset, an alternative, distributed, scalable approach has also been proposed capable of being applied to online lod triplestores.
- *The application should show the use of the very large, mixed quality data set.*
  The application can cope with the varying quality of BTC dataset sources and demonstrates a case of qualitty ranking by entity popularity facilitating the user to select useful info.
- *The application should either function in real-time.*
  The proposed application requires pre-computation for the identification of the underlying URIs, but it is realized real-time, since it has a end-user part that functions in real-time.