Editor: Carole Goble • carole.goble@manchester.ac.uk

Cool URIs and Dynamic Data

Robert Sanderson and Herbert Van de Sompel • *Los Alamos National Laboratory*

Linked datasets contain descriptions that change over time. Applications that leverage linked data must be aware of these change dynamics to deliver accurate services. Here, the authors highlight important challenges that are involved in dealing with change and review possible solutions.

In keeping with the tradition of the Linked Data department, a respectful nod to Tim Berners-Lee is in order. His "Cool URIs Don't Change" musings[1] have inspired website managers to devise strategies aimed at maintaining the existence of HTTP URIs over time, even amidst technical and organizational change. The *Architecture of the World Wide Web*[2] further emphasizes the importance of keeping URIs unaltered when the content available from them changes, an approach with obvious benefits. For one thing, modifying a URI in lockstep with changing content would yield an unmanageable number of broken links. Moreover, a Web architecture that at any moment in time only supports access to a resource's current content — and lacks even the notion of access to prior content — is simpler, thus positively affecting its adoption. Nevertheless, there are many information use cases for which a Web that exists in the eternal now isn't acceptable. The many offerings of content management systems with elaborate versioning mechanisms, and the existence of rapidly expanding Web archives worldwide, illustrate both information publishers' and consumers' desire to store and access current and prior content.

Changing content available from the same URI is just one illustration of the Web's dynamics. In addition, resources are continuously created, moved and deleted, linked and unlinked. The Web is in permanent motion, and its dynamics have been the subject of extensive research ever since its inception. In many cases, the focus is on optimizing crawl strategies for search engines as a means of allowing their indexes to reflect the Web's ever-evolving state in a timely manner. However, the research findings are diverse and substantial, informing the implementation of many other practical applications.

If keeping a finger on the pulse of evolving webpages is deemed important, doing the same for linked data is at least equally crucial, if not more so. Linked data is frequently crawled and merged in triple stores to drive applications. Understanding and being able to deal with the constituent datasets' particular rate of change is essential for delivering accurate responses to queries across the multisource information kept in the triple store. For example, the prices and product descriptions of many major online retailers, described using the GoodRelations vocabulary (www.heppnetz.de/projects/goodrelations/), will change less frequently than discussion about those products, marked up with the Semantically Interlinked Online Communities ontology (http://sioc-project.org). In turn, these product descriptions and discussions might refer to the language they're written in from the lingvoj (www.lingvoj.org) or lexvo (www.lexvo.org) datasets, which change at a glacial rate in comparison. Yet, keeping all three up to date is important for answering questions about the demographics of market feeling toward products.

Despite its importance, the current understanding of change dynamics for the *data Web* is far less mature than for *document Web*, given

 Published by the IEEE Computer Society 1089-7801/12/$31.00 © 2012 IEEE

that research into this newer realm has only just started. So far, few concerted efforts aim at actively archiving linked datasets and making such archives as easily accessible as traditional Web archives. Consequently, finding datasets where the information matches up temporally is challenging. Overall, dealing with change in linked data remains hard — so hard that a recent study found that as many as 70 percent of surveyed applications hadn't even attempted it.[3] For many important needs related to changing data, implementation patterns or best practices remain elusive. These include how to publish, access, archive, store, and query data that's in flux, as well as and how to detect, predict, and communicate change.

Publishing and Consuming Versions

An appealing pattern has emerged regarding publishing and accessing versions of linked data descriptions as they evolve over time. Take, for example, an RDF description for the city of Berlin, hypothetically available from `http://example.org/data/Berlin`. As indicated in the introduction, the current description is always available from that "generic" URI.

In addition, each version of the RDF description could receive its own "version" URI: `http://example.org/v1/data/Berlin`, `http://example.org/v2/data/Berlin`, and so forth. Each of these descriptions would be expressed in terms of the constituent resources' version-less URIs, following the recommendations of the *Architecture of the World Wide Web*. Although both current and prior descriptions are available, two problems remain: how to navigate between versions and how to understand which version was current at a specific point in time.

One approach is to convey this information in the RDF descriptions available from the generic and version URIs.[4] A linked data publisher would do this by including a list of all corresponding version URIs alongside the description of Berlin. In addition, a *validity interval* would be expressed for each version URI, letting a client determine which version matches its temporal preference. This approach introduces complications when descriptions contain links to external resources for which a client must also determine and retrieve an appropriate temporal version. Also, as new versions are added, the list of version URIs in each description will necessarily need to be updated to remain complete. This requires modifying archived descriptions potentially no longer under the original system's control.

A second approach, emerging from the Memento "time travel for the Web" project, which we helped to initiate, leverages the existing HTTP infrastructure to navigate to an appropriate version.[5] The generic URI still serves as a hub, but an HTTP client interested in prior versions expresses a date/time preference in a special-purpose `Accept-Datetime` HTTP header when accessing the URI. Using existing content-negotiation mechanisms applied to time rather than, for example, format, a client would retrieve the version that was current at the time conveyed in the header. Memento also supports retrieving a list of all known versions and their temporal validity interval — termed a *TimeMap* — from a completely separate URI, thereby removing the need for updating any archived descriptions.

Memento is applicable to both documents and data because it is defined in terms of HTTP, and thus effectively allows for following hyperlinks and linked data references subject to time. As a result, clients can also navigate versioned vocabularies subject to time, allowing for the correct interpretation of relationships and properties in RDF descriptions, if time-stamped ontology versions are available. Memento further allows current and prior versions to exist on different systems and introduces powerful capabilities for gathering temporal snapshots of interlinked datasets using a basic HTTP "follow-your-nose" approach.[6] At the time of writing, one of the largest linked datasets, DBpedia (http://dbpedia.org), supported Memento time travel; prior versions of descriptions are available from a linked data archive we operate (http://dbpedia.mementodepot.org).

The previously mentioned validity interval deserves further attention owing to how versions of linked data descriptions are created and maintained. Many existing linked datasets become available via recurrent dumps; DBpedia is a good example. In this case, each description is temporally valid at the moment the dump is generated, but determining the temporal interval over which it's valid is a much more subtle problem. If, as is the case with DBpedia, consecutive dumps of a dataset are different but no ongoing changes are made to the data in the period between the dumps, then this period's duration is the validity interval. However, if the data changes continuously, as is the case with DBpedia Live (http://live.dbpedia.org), the validity interval differs per description, and it can't be determined from the dump-file approach.

Web-based content management systems that record and provide access to every version of a resource via a version-specific URI, such as some DataWiki implementations, are well suited to store linked datasets that evolve in this manner. Moreover, recording the creation time stamp as metadata about each version ensures that the validity interval for each is well understood. This proactive approach to minting versions

contrasts with one that relies on third parties to recurrently crawl a dynamic dataset to obtain and store the evolving RDF descriptions. In this latter approach, determining the validity interval is impossible because only discreet observations are available, rather than the descriptions' entire history. When a consuming application merges descriptions with uncertain temporal validity that potentially originate in different datasets and uses them for querying and inference, the outcomes might be unreliable.

An open challenge in the linked data environment is how to deal with queries that involve time — either searching a single description across multiple versions (and hence times) or searching across multiple descriptions at a single time other than the present. Google Freebase's Metaweb Query Language (MQL; http://wiki.freebase.com/wiki/MQL_Manual) allows for searching at a particular point in the past and accesses only the resources that were current at that point; however, this isn't available in the standard SPARQL query language. Searching between versions of the same document also isn't part of the language, because the RDF data model doesn't have a notion of the serialized document, just the information that it carries. IBM's DB2 product (http://www.ibm.com/software/data/db2/), as of version 10, also supports RDF and a "time travel" feature whereby users can query the database at a given point in time.

Understanding, Detecting, and Communicating Change

The entire linked data ecosystem's change dynamics aren't yet well understood, other than its rapid growth, illustrated by the frequently presented *linked data cloud* diagrams. Work in this realm is starting to emerge, however, both at the research and infrastructural levels. An example of the former is the pioneering work of the Digital Enterprise Research Institute (DERI) group that analyzed various aspects of change across 24 weekly snapshots of the 7-hop neighborhood of Tim Berners-Lee's Friend-of-a-Friend file.[7] They tried to assess whether linked data's change process can be modeled as a Poisson process, as with the document Web, but their findings were inconclusive. More recently, efforts have started to collect reference datasets that can support studying change dynamics. For example, the Dynamic Linked Data Observatory (http://swse.deri.org/dyldo/) intends to make the results of recurrent crawls of a representative portion of the linked data cloud publicly available. Similarly, the Web Data Commons (http://webdatacommons.org) publishes datasets that result from extracting structured data, such as RDFa, from the Common Crawl datasets that themselves result from recurrent crawls of the document Web.

Being able to understand and characterize change is important for many practical reasons. When applications locally store descriptions that originate from various remote datasets, awareness of the changes that such descriptions undergo at their origin allows for timely updating, and hence delivering services based on fresh information. Also, an understanding of a dataset's pace of change can inform a decision as to whether an application will cache its content locally, or remotely query it on the fly. Generally speaking, change detection is important for dataset synchronization, smart caching, link maintenance, and vocabulary evolution.[8]

Widely accepted patterns to support change detection and communication have yet to be established, but outlines of possible approaches are emerging in three areas, via the Dataset Dynamics effort among others (www.w3.org/wiki/DatasetDynamics). The first area addresses the description of change at the dataset level. The DatasetDynamics (dady; http://vocab.deri.ie/dady/) vocabulary, an extension to the Vocabulary of Interlinked datasets (VoiD; www.w3.org/TR/void/), lets publishers characterize the pace of change using coarse qualifiers such as "HighFrequentUpdates" and "IrregularUpdates." The Semantic Sitemaps extension to the widely used Sitemap protocol provides similar functionality (http://sw.deri.org/2007/07/sitemapextension/), but is more focused on supporting the discovery of different access points for a dataset, such as the location of a dump or a SPARQL end point.

The second area is concerned with change notifications that could pertain to individual RDF descriptions, entire datasets, or even query results. The publish-subscribe paradigm is commonly mentioned as an implementation strategy. Using this approach, a dataset owner publishes change notifications to a channel, and interested consumers subscribe to the channel to remain aware of, and act on, changes. The Atom-based PubSubHubbub (http://code.google.com/p/pubsubhubbub/) is regarded as a candidate technology and has been applied in SparqlPuSH,[9] an approach to send change notifications pertaining to registered SPARQL queries. Recently, the ResourceSync effort — on which the US National Information Standardization Organization (NISO) and the Open Archives Initiative (OAI) collaborate — has looked into using XMPP PubSub[10] as the protocol to communicate change notifications for resources on both the document and data Web.

The third, and probably least understood, area addresses how to describe the change itself. Here, questions related to granularity arise. For example, when RDF descriptions are concerned, should change be described at the level of a single updated RDF statement, as in DBpedia Live, or as deltas between two

consecutive versions of the RDF description? Or, should the entire RDF description be exchanged every time it's modified? Making choices with this regard involves considerations related to payload size, bandwidth, update pace, and whether a change description approach can generically be applied for both data Web and document Web resources.

Here, we've provided a perspective on various aspects related to linked data and change. As is the case with handling change in general, managing change in a linked data environment turns out to be hard. However, the linked data community is actively pursuing new insights about dataset dynamics, and patterns are emerging that support dealing with data as it evolves. Change, as they say, is the only thing that remains the same.

References

1. T. Berners-Lee, "Cool URIs Don't Change," *W3C Style*, 1998; www.w3.org/Provider/Style/URI.
2. I. Jacobs and N. Walsh, *Architecture of the World Wide Web, Volume One*, W3C recommendation, 15 Dec. 2004; www.w3.org/TR/webarch/.
3. B. Heitmann et al., "An Empirically Grounded Conceptual Architecture for Applications on the Web of Data," *IEEE Trans. Systems, Man, and Cybernetics – Part C: Applications and Reviews*, vol. 42, no. 1, 2012, pp. 51–60; doi:10.1109/TSMCC.2011.2145370.
4. J. Tenison, "Versioning (UK Government) Linked Data," blog, 27 Feb. 2010; www.jenitennison.com/blog/node/141.
5. H. Van de Sompel, M. Nelson, and R. Sanderson, "HTTP Framework for Time-Based Access to Resource States – Memento," IETF Internet draft, work in progress, Dec. 2011; https://datatracker.ietf.org/doc/draft-vandesompel-memento/.
6. H. Van de Sompel et al., "An HTTP-Based Versioning Mechanism for Linked Data," *Proc. 3rd Workshop Linked Data on the Web*, 2010; http://arxiv.org/abs/1003.3661.
7. J. Umbrich et al., "Towards Dataset Dynamics: Change Frequency of Linked Open Data Sources," *Proc. 3rd Workshop Linked Data on the Web*, 2010; http://ceur-ws.org/Vol-628/ldow2010_paper12.pdf.
8. J. Umbrich, B. Villazon-Terrazas, and M. Hausenblas, "Dataset Dynamics Compendium: A Comparative Study," *Proc. 1st Int'l Workshop Consuming Linked Data*, 2010; http://ceur-ws.org/Vol-665/UmbrichEtAl_COLD2010.pdf.
9. A. Passant and P.N. Mendes, "sparqlPuSH: Proactive Notification of Data Updates in RDF Stores Using PubSubHubbub," *Proc. 6th Workshop Scripting and Development for the Semantic Web*, 2010; http://ceur-ws.org/Vol-699/Paper6.pdf.
10. P. Millard, P. Saint-Andre, and R. Meijer, "XEP-0060: Publish-Subscribe," XMPP Standards Foundation draft standard, Jul. 2010; http://xmpp.org/extensions/xep-0060.html.

Robert Sanderson is a scientist at Los Alamos National Laboratory. His research interests include interoperability architectures for scholarly communication and the application of high-performance computing and machine learning for digital preservation. Sanderson has a PhD in medieval French from the University of Liverpool. Contact him at rsanderson@lanl.gov.

Herbert Van de Sompel is a scientist at Los Alamos National Laboratory. His research interests include information interoperability for digital scholarship and the transformation of research communication. Van de Sompel has a PhD in communication science from Ghent University. Contact him at herbertv@lanl.gov.