Managing the Life-Cycle of Linked Data with the LOD2 Stack

Sören Auer, Sebastian Tramp, Bert van Nuffelen, Robert Isele, Jens Lehmann, Lorenz Bühmann, Christian Dirschl, Pablo N. Mendes, Hugh Williams, Orri Erling, Michael Hausenblas

LOD2 Project**, c/o Universität Leipzig, Postfach 100920, 04009 Leipzig, Germany
http://lod2.eu

Abstract. The LOD2 Stack is an integrated distribution of aligned tools which support the whole life cycle of Linked Data from extraction, authoring/creation via enrichment, interlinking, fusing to maintenance. The LOD2 Stack comprises new and substantially extended existing tools from the LOD2 project partners and third parties. The stack is designed to be versatile; for all functionality we define clear interfaces, which enable the plugging in of alternative third-party implementations. The architecture of the LOD2 Stack is based on three pillars: **(1)** Software integration and deployment using the Debian packaging system. **(2)** Use of a central SPARQL endpoint and standardized vocabularies for knowledge base access and integration between the different tools of the LOD2 Stack. **(3)** Integration of the LOD2 Stack user interfaces based on REST enabled Web Applications. These three pillars comprise the methodological and technological framework for integrating the very heterogeneous LOD2 Stack components into a consistent framework. In this article we describe these pillars in more detail and give an overview of the individual LOD2 Stack components. The article also includes a description of a real-world usage scenario in the publishing domain.

Keywords: Linked Data, application integration, provenance

1 Introduction

The LOD2 Stack is an integrated distribution of aligned tools which support the whole life cycle of Linked Data from extraction, authoring/creation via enrichment, interlinking, fusing to maintenance. The LOD2 Stack comprises new and substantially extended existing tools from the LOD2 partners and third parties. The major components of the LOD2 Stack are open-source in order to facilitate wide deployment and scale to knowledge bases with billions of triples and large numbers of concurrent users. Through an agile, iterative software development

** The research leading to these results has received funding under the European Commission's Seventh Framework Programme (FP7/2007–2013) from ICT grant agreement LOD2, no. 257943.

approach, we aim at ensuring that the stack fulfills a broad set of user requirements and thus facilitates the transition to a Web of Data. The stack is designed to be versatile; for all functionality we define clear interfaces, which enable the plugging in of alternative third-party implementations. We also plan a stack configurer, which enables potential users to create their own personalized version of the LOD2 Stack, which contains only those functions relevant for their usage scenario.

In order to fulfill these requirements, the architecture of the LOD2 Stack is based on three pillars:

- *Software integration and deployment using the Debian packaging system.* The Debian packaging system is one of the most widely used packaging and deployment infrastructures and facilitates packaging and integration as well as maintenance of dependencies between the various LOD2 Stack components. Using the Debian system also allows to facilitate the deployment of the LOD2 Stack on individual servers, cloud or virtualization infrastructures.
- *Use of a central SPARQL endpoint and standardized vocabularies for knowledge base access and integration between different tools.* All components of the LOD2 Stack access this central knowledge base repository and write their findings back to it. In order for other tools to make sense out of the output of a certain component, it is important to define vocabularies for each stage of the Linked Data life-cycle.
- *Integration of the LOD2 Stack user interfaces based on REST enabled Web Applications.* Currently, the user interfaces of the various tools are technologically and methodologically quite heterogeneous. We do not resolve this heterogeneity, since each tool's UI is specifically tailored for a certain purpose. Instead, we develop a common entry point for accessing the LOD2 Stack UI, which then forwards a user to a specific UI component provided by a certain tool in order to complete a certain task.

These three pillars comprise the methodological and technological framework for integrating the very heterogeneous LOD2 Stack components into a consistent framework. This article is structured as follows: After briefly introducing the linked data life-cycle in Section 2, we describe these pillars in more detail (Section 3). We describe a real-world use-case for the Stack in Section 4 and conclude with an outlook on future work in Section 5.

2 The Linked Data Life-Cycle

The different stages of the Linked Data life-cycle are depicted in Figure 1. They include:

Storage. RDF Data Management is still more challenging than relational Data Management. We aim to close this performance gap by employing column-store technology, dynamic query optimization, adaptive caching of joins, optimized graph processing and cluster/cloud scalability.

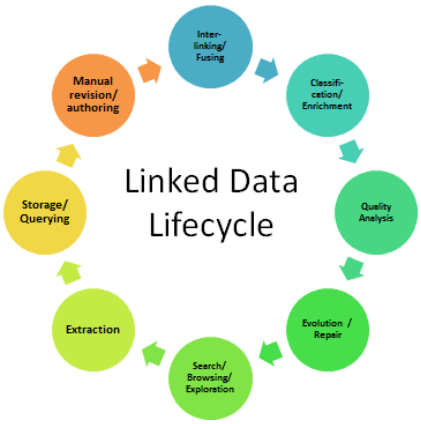

Fig. 1. Stages of the Linked Data life-cycle supported by the LOD2 Stack.

Authoring. LOD2 facilitates the authoring of rich semantic knowledge bases, by leveraging Semantic Wiki technology, the WYSIWYM paradigm (What You See Is What You Mean) and distributed social, semantic collaboration and networking techniques.

Interlinking. Creating and maintaining links in a (semi-)automated fashion is still a major challenge and crucial for establishing coherence and facilitating data integration. We seek linking approaches yielding high precision and recall, which configure themselves automatically or based on end-user feedback.

Classification. Linked Data on the Web is mainly raw instance data. For data integration, fusion, search and many other applications, however, we need this raw instance data to be linked and integrated with upper level ontologies.

Quality. The quality of content on the Data Web varies, as the quality of content on the document web varies. LOD2 develops techniques to help assessing quality based on characteristics such as provenance, context, coverage or structure.

Evolution/Repair. Data on the Web is dynamic. We need to facilitate the evolution of data while keeping things stable. Changes and modifications to knowledge bases, vocabularies and ontologies should be transparent and observable. LOD2 also develops methods to spot problems in knowledge bases and to automatically suggest repair strategies.

Search/Browsing/Exploration. For many users, the Data Web is still invisible below the surface. LOD2 develops search, browsing, exploration and visualization techniques for different kinds of Linked Data (i.e. spatial, temporal, statistical), which make the Data Web sensible for real users.

These life-cycle stages, however, should not be tackled in isolation, but by investigating methods which facilitate a mutual fertilization of approaches developed to solve these challenges. Examples for such mutual fertilization between approaches include:

- The detection of mappings on the schema level, for example, will directly affect instance level matching and vice versa.
- Ontology schema mismatches between knowledge bases can be compensated for by learning which concepts of one are equivalent to which concepts of another knowledge base.
- Feedback and input from end users (e.g. regarding instance or schema level mappings) can be taken as training input (i.e. as positive or negative examples) for machine learning techniques in order to perform inductive reasoning on larger knowledge bases, whose results can again be assessed by end users for iterative refinement.
- Semantically enriched knowledge bases improve the detection of inconsistencies and modelling problems, which in turn results in benefits for interlinking, fusion, and classification.
- The querying performance of RDF data management directly affects all other components, and the nature of queries issued by the components affects RDF data management.

As a result of such interdependence, we should pursue the establishment of an improvement cycle for knowledge bases on the Data Web. The improvement of a knowledge base with regard to one aspect (e.g. a new alignment with another interlinking hub) triggers a number of possible further improvements (e.g. additional instance matches).

The challenge is to develop techniques which allow exploitation of these mutual fertilizations in the distributed medium Web of Data. One possibility is that various algorithms make use of shared vocabularies for publishing results of mapping, merging, repair or enrichment steps. After one service published its new findings in one of these commonly understood vocabularies, notification mechanisms (such as *Semantic Pingback* [15]) can notify relevant other services (which subscribed to updates for this particular data domain), or the original data publisher, that new improvement suggestions are available. Given proper management of provenance information, improvement suggestions can later (after acceptance by the publisher) become part of the original dataset.

3 Integrating Heterogeneous Tools into the LOD2 Stack

The LOD2 Stack serves two main purposes. Firstly, the aim is to ease the distribution and installation of tools and software components that support the Linked Data publication cycle. As a distribution platform, we have chosen the well established Debian packaging format. The second aim is to smoothen the information flow between the different components to enhance the end-user experience by a more harmonized look-and-feel.

3.1 Deployment management leveraging Debian packaging

In the *Debian package management system* [12], software is distributed in architecture-specific binary packages and architecture-independent source code packages. A

Debian software package comprises two types of content: **(1)** control information (incl. metadata) of that package, and **(2)** the software itself.

The control information of a Debian package will be indexed and merged together with all other control information from other packages available for the system. This information consists of descriptions and attributes for:

(a) The software itself (e.g. licenses, repository links, name, tagline, ...),
(b) Its relation to other packages (dependencies and recommendations),
(c) The authors of the software (name, email, home pages), and
(d) The deployment process (where to install, pre and post install instructions).

The most important part of this control information is its relations to other software. This allows the deployment of a complete stack of software with one action. The following dependency relations are commonly used in the control information:

Depends: This declares an absolute dependency. A package will not be configured unless all of the packages listed in its Depends field have been correctly configured. The Depends field should be used if the depended-on package is required for the depending package to provide a significant amount of functionality. The Depends field should also be used if the install instructions require the package to be present in order to run.

Recommends: This declares a strong, but not absolute, dependency. The Recommends field should list packages that would be found together with this one in all but unusual installations.

Suggests: This is used to declare that one package may be more useful with one or more others. Using this field tells the packaging system and the user that the listed packages are related to this one and can perhaps enhance its usefulness, but that installing this one without them is perfectly reasonable.

Enhances: This field is similar to Suggests but works in the opposite direction. It is used to declare that a package can enhance the functionality of another package.

Conflicts: When one binary package declares a conflict with another using a Conflicts field, dpkg will refuse to allow them to be installed on the system at the same time. If one package is to be installed, the other must be removed first.

All of these relations may restrict their applicability to particular versions of each named package (the relations allowed are <<, <=, =, >= and >>). This is useful in forcing the upgrade of a complete software stack. In addition to this, dependency relations can be set to a list of alternative packages. In such a case, if any one of the alternative packages is installed, that part of the dependency is considered to be satisfied. This is useful if the software depends on a specific functionality on the system instead of a concrete package (e.g. a mail server or a web server). Another use case of alternative lists are meta-packages. A meta-package is a package which does not contain any files or data to be installed. Instead, it has dependencies on other (lists of) packages.

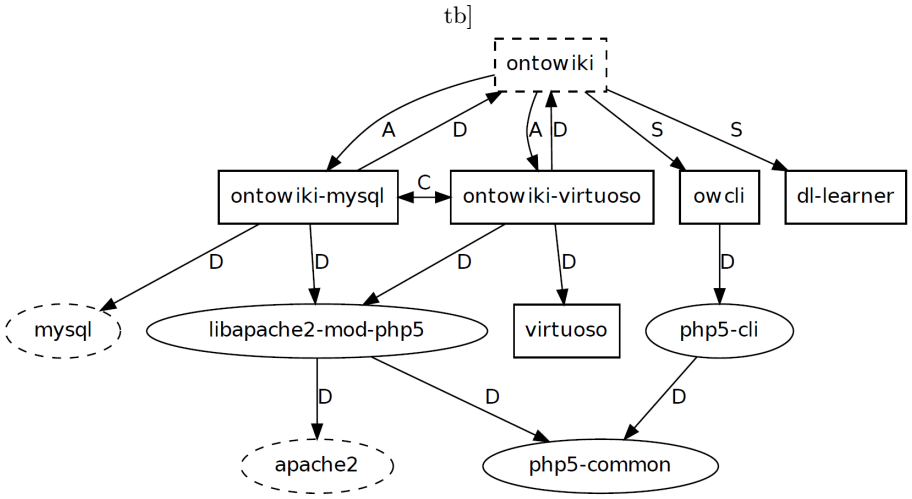

Fig. 2. Example DEB-package dependency tree (OntoWiki). Some explanation: Boxes are part of the LOD2 Stack, Ellipses are part of the Debian/Ubuntu base system, Dashed forms are meta-packages, Relations: Depends (D), Depends alternative list (A), Conflicts (C) and Suggests (S).

Example of meta-packaging: OntoWiki. To build an appropriate package structure, the first step is to inspect the manual deployment of the software, its variants and the dependencies of these variants. *OntoWiki* is a browser-based collaboration and exploration tool as well as an application for linked data publication. There are two clusters of dependencies: the runtime environment and the backend. Since OntoWiki is developed in the scripting language *PHP*, it's architecture-independent but needs a web server running PHP. More specifically, OntoWiki needs PHP5 running as an Apache 2 module. OntoWiki currently supports two different back-ends which can be used to store and query RDF data: *Virtuoso* and MySQL. Virtuoso is also part of the LOD2 Stack while *MySQL* is a standard package in all Debian-based systems. In addition to OntoWiki, the user can use the OntoWiki command line client *owcli* and the *DL-Learner* from the LOD2 Stack to enhance its functionality.

The dependency tree (depicted in Figure 2) is far from being complete, since every component also depends on libraries and additional software which is omitted here. Given this background information, we can start to plan the packaging. We assume that users either use MySQL or Virtuoso as a backend on a server, so the first decision is to split this functionality into two packages: `ontowiki-mysql` and `ontowiki-virtuoso`. These two packages are abstracted by the meta-package `ontowiki`, which requires either `ontowiki-mysql` or `ontowiki-virtuoso`, and which can be used by other LOD2 Stack packages to require OntoWiki. Since both the MySQL backend and the Virtuoso backend

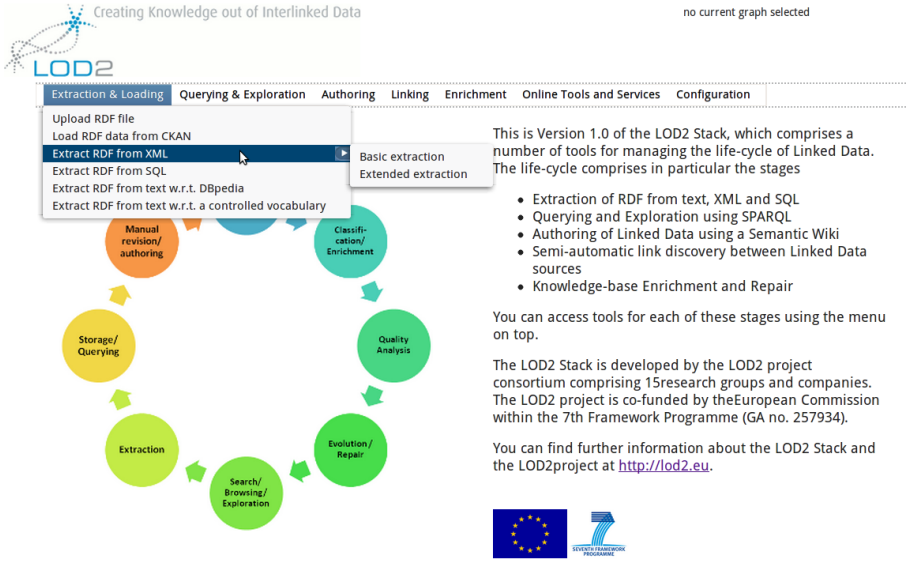

Fig. 3. The LOD2 Stack demonstrator is an interface to explore and use all the different stack tools in an integrated way.

version use the same system resources, we need to declare them as conflicting packages.

Installing the LOD2 Stack The LOD2 Stack is available at `http://stack.lod2.eu`. Our reference OS is *Ubuntu 12.04 LTS*. Most of the components run on old or more recent releases without a problem. In general, deploying the LOD2 software stack or parts of it is simple. There are only two steps to execute in order to install LOD2 Stack software: **(1)** Add the LOD2 Stack package repository to the system's repository list and update the repository index. **(2)** Install desired software packages by using a graphical or text-based package management application. The procedure can be executed using graphical front-ends like Synaptic. Using the command line the LOD2 Stack installation is performed as follows[1]:

```
# download the repository package
wget http://stack.lod2.eu/lod2repository_current_all.deb
# install the repository package
sudo dpkg -i lod2repository_current_all.deb
# update the repository database
sudo apt-get update

# lod2demo is a meta root package that installs all LOD2 components
sudo apt-get lod2demo
```

[1] More information, tutorials and FAQs can be found at `http://wiki.lod2.eu`.

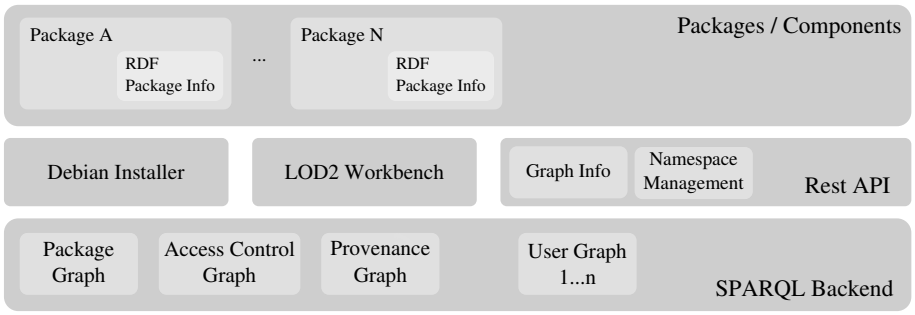

Fig. 4. Basic architecture of a local LOD2 Stack.

3.2 Tool integration based on central SPARQL endpoint, WebID and standardized vocabularies

The basic architecture of a local LOD2 Stack installation is depicted in Figure 4. All components in the LOD2 Stack act upon RDF data and are able to communicate via SPARQL with the central system-wide RDF quad store (i.e. SPARQL backend). This quad store (Openlink Virtuoso) manages user graphs (knowledge bases) as well as a set of specific system graphs where the behaviour and status of the overall system is described. The following system graphs are currently in use:

Package Graph: In addition to the standard Debian package content, each LOD2 Stack package consists of a RDF package info which describes the following details:
- The basic package description, e.g. labels, dates, maintainer info (this is basically DOAP data and redundant to the classic Debian control file)
- Pointers to the place where the application is available (e.g. the menu entry in the LOD2 Stack workbench)
- A list of capabilities of the packed software (e.g. resource linking, RDB extraction). These capabilities are part of a controlled vocabulary. The terms are used as pointers for provenance logging, access control definition and a future capability browser of the LOD2 workbench.

Upon installation, the package info is automatically added to the package graph to allow the workbench / demonstrator to query which applications are available and what is the user able to do with them.

Access Control Graph: This system graph is related to WebID[2] authentication and describes which users are able to use which capabilities and have access to which graphs. The default state of this graph contains no restrictions, but could be used to restrict certain WebIDs to specific capabilities. Currently, only OntoWiki takes this graph into account and the access control definition is based on the WebAccessControl schema[3].

[2] `http://www.w3.org/wiki/WebID`
[3] `http://www.w3.org/wiki/WebAccessControl`

Tool	Category	Supported Stages
Apache Stanbol [3]	NLP Middleware Server	Extraction
DBpedia Spotlight [10]	Entity Recognition and Linking	Extraction
D2RQ [2]	RDB2RDF Mapping	Extraction
DL-Learner [6,7,9]	Machine Learning in OWL	Schema Enrichment
OntoWiki [1]	Generic Data Wiki	Authoring, Exploration
ORE [8]	Knowledge Base Debugging	Repair
PoolParty [14]	SKOS Taxonomy Editor	Authoring, Exploration
Sig.ma EE [16]	Data Browser	Search, Exploration
Sieve [11]	Quality Assessment and Fusion	Quality, Repair
SILK [5]	Linking Workbench	Interlinking
LIMES [13]	Linking Workbench	Interlinking
Virtuoso [4]	Hybrid RDBMS/Graph Column Store	Storage / Querying
Valiant	XML2RDF transformation	Extraction

Table 1. Overview on LOD2 Stack components.

Provenance Graph: Each software package is able to log system wide provenance information to reflect the evolution of a certain knowledge base. Different ontologies are developed for that use-case. To keep the context of the LOD2 Stack, we use the controlled capability vocabulary as reference points.

In addition to the SPARQL protocol endpoint, application packages can use a set of APIs which allow queries and manipulation currently not available with SPARQL alone (e.g. fetching graph information and manipulating namespaces).

Two authorized administration tools are allowed to manipulate the package and access control graphs:

- The Debian system installer application automatically adds and removes package descriptions during install / upgrade and remove operations.
- The LOD2 Workbench (Demonstrator) is able to manipulate the access control graph.

All other packages are able to use the APIs as well as to create, update and delete knowledge bases. Table 1 gives an overview on the current LOD2 Stack components in alphabetic order. In the following, we give a brief summary on some of the most important packages.

Apache Stanbol: Apache Stanbol (currently in incubation) is an open source modular software stack and reusable set of components for semantic content management. Apache Stanbol components are meant to be accessed over RESTful interfaces to provide semantic services for content management. Thus, one application is to extend traditional content management systems with (internal or external) semantic services. In the context of the LOD2 Stack, Apache Stanbol can be used for NLP services which rely on the stack internal knowledge bases, such as named entity recognition and

text classification. Apache Stanbol itself is the result of the EU-funded IKS project[4].

DBpedia Spotlight: DBpedia Spotlight is a tool for automatically annotating mentions of DBpedia resources in text, providing a solution for linking unstructured information sources to the Linked Open Data cloud through DBpedia. DBpedia Spotlight recognizes that names of concepts or entities have been mentioned (e.g. "Michael Jordan"), and subsequently matches these names to unique identifiers (e.g. `dbpedia:Michael_I._Jordan`, the machine learning professor or `dbpedia:Michael_Jordan`, the basketball player). Besides common entity classes such as People, Locations and Organisations, DBpedia Spotlight also spots concepts from any of the 320 classes in the DBpedia Ontology. DBpedia Spotlight is also supported in Apache Stanbol, and thus can be combined with other NLP tools in a modular fashion.

D2RQ: D2RQ[5] is a system for integrating relational databases (RDBMS) in RDF-based data integration workflows. The D2RQ software was originally developed starting in 2004 at FU Berlin. Since 2007, development has been shared between DERI, FU Berlin and other contributors. It allows querying a non-RDF database using SPARQL, accessing the content of the database as Linked Data over the Web, creating custom dumps of the database in RDF formats for loading into an RDF store, and accessing information in a non-RDF database using the *Apache Jena API*. D2RQ has been downloaded more than 13,000 times, and powers roughly 100 public Linked Data sites around the Web. D2RQ supports RDBMSs from all major vendors, including *Oracle*, *MySQL*, *PostgreSQL*, *SQL Server*, *HSQLDB*, and *Interbase/Firebird*. Current work focuses on extending D2RQ and making it compliant with *W3C's R2RML* and *Direct Mapping* standards[6].

DL-Learner: The DL-Learner framework provides a set of (semi)supervised machine learning algorithms for Semantic Web knowledge bases, specifically for OWL ontologies and SPARQL endpoints. The goal of DL-Learner is to support knowledge engineers in constructing knowledge and learning about the data they created, by generating axioms and concept descriptions which fit the underlying data.

ORE: The ORE (Ontology Repair and Enrichment) tool allows knowledge engineers to improve an OWL ontology or SPARQL endpoint backed knowledge base by fixing logical errors and making suggestions for adding further axioms to it. ORE uses state-of-the-art methods to detect errors and highlight the most likely sources for the problems. To harmonise schema and data in the knowledge base, algorithms of the DL-Learner framework are integrated.

OntoWiki: OntoWiki is a *PHP5* / Zend-based Semantic Web application for collaborative knowledge base editing. It facilitates the visual presentation of a knowledge base as an information map, with different views of instance data. It enables intuitive authoring of semantic content, with an inline editing mode for editing RDF content, similar to WYSIWYG for text documents.

[4] `http://iks-project.eu`
[5] `http://d2rq.org/`
[6] `http://www.w3.org/2001/sw/rdb2rdf/`

PoolParty PoolParty is a tool to create and maintain multilingual *SKOS* (Simple Knowledge Organisation System) thesauri, aiming to be easy to use for people without a Semantic Web background or special technical skills. PoolParty is written in Java and uses the *SAIL API*, whereby it can be utilized with various triple stores, which allows for flexibility in terms of performance and scalability. Thesaurus management itself (viewing, creating and editing SKOS concepts and their relationships) can be done in an *AJAX* front-end based on the Yahoo User Interface (YUI) library.

Sig.ma EE Sig.ma is an on-the-fly Web of Data mashup creation interface. Sig.ma Enterprise Edition (Sig.ma EE) is a standalone, deployable, customizable version of Sig.ma. Sig.ma EE is deployed as a web application and will perform on-the-fly data integration from both local LOD2 Stack internal data sources and remote services.

Sieve Sieve includes a Quality Assessment module and a Data Fusion module. The quality of Linked Data sources on the Web is varies widely, as values may be out of date, incomplete or incorrect. Moreover, data sources may provide conflicting values for a single real-world object. Sieve's Quality Assessment module leverages user-selected metadata as quality indicators to produce quality assessment scores through user-configured scoring functions. The Data Fusion module is able to use quality scores in order to perform user-configurable conflict resolution tasks.

Silk The Silk Link Discovery Framework supports data publishers in setting explicit links between two datasets. Using the declarative Silk - Link Specification Language (Silk-LSL), developers can specify which types of RDF links should be discovered between data sources as well as which conditions data items must fulfill in order to be interlinked. These link conditions may combine various similarity metrics and can take the graph around a data item into account using an RDF path language.

LIMES LIMES is a link discovery framework for the Web of Data. It implements time-efficient approaches for large-scale link discovery based on the characteristics of metric spaces. It is easily configurable via a web interface. It can also be downloaded as a standalone tool for carrying out link discovery locally. In addition, the Colanut GUI implements mechanisms for the automatic suggestion of link configurations.

Virtuoso Virtuoso is an enterprise grade multi-model data server. It delivers a platform agnostic solution for data management, access, and integration. Virtuoso provides a fast quad store as well as a *SPARQL* endpoint with WebID support.

Valiant Valiant is an extraction/transformation tool that uses XSLT to transform XML documents into RDF. The tool can access data from the file system or a WebDAV repository. It outputs the resulting RDF to disk, WebDAV or directly to an RDF store. For each input document a new graph is created.

3.3 REST integration of user interfaces

Many of the components come with their own user interface. For example, the Silk Workbench is a user interface for the Silk linking engine. This workbench supports the creation of linking specifications, executing them and improving them using the feedback from the user on the created links. With the OntoWiki linked data browsing and authoring tool, a user can browse and update information in a knowledge base. By using both tools together, the user the ability to study the input sources' content structure and to create links between them.

Many stack components request similar information from the user. For example, selecting the graph of interest. To provide the end-user the feeling of a harmonized single application, we develop supportive REST-based WebAPIs. These APIs offer a common application view of the LOD2 Stack. The more tools support this API, the more harmonized and integrated the end-user experience gets. Currently, the LOD2 Stack WebAPI consists of:

- *Graph management*: The set of graphs is not easy to maintain. SPARQL does not support retrieval of all graphs (including empty ones). The only possible query which selects all graphs that have at least one triple is performance wise quite costly: `SELECT DISTINCT ?g WHERE GRAPH ?g ?s ?p ?o` The WebAPI also standardizes some meta information like *being a system graph*. When LOD2 Stack components use this common graph management WebAPI, the end-user obtains a uniform look-and-feel with respect to graph management.
- *Prefix management*: To make RDF resources more readable, prefixes are used to abbreviate URI namespaces. Typically, each application manages its own namespace mapping. Using this REST API, a central namespace mapping is maintained, thus producing consistency among stack components. The end-user is freed from updating the individual component mappings. Moreover, an update in one component is immediately available to another.

In addition to creating supportive REST-based APIs, the LOD2 Stack stimulates the component owners to open up their components using REST based WebAPIs. For example, the semantic-spatial browser, a UI tool that visualizes RDF data containing geospatial information on a map, is entirely configurable by parameters encoded within its invocation URL. This makes it easy to integrate into (third party) applications.

3.4 Enlarging the LOD volume and facilitating dataset discovery

All the above effort to improve the software support for Linked Data publishing must have an effect in the daily practice of Linked Data publishing. For that reason the LOD2 project collaborates with data providers. One such data provider is the LOD2 partner *Wolters Kluwer*. Other collaborations include the *European Commission DG INFSO*, with its *Digital Agenda Scoreboard*, and the *National Statistical Office of Serbia*. Both improvements to the tools and data are returned to the public.

To ease reuse, data must be easily found. Therefore, we enhanced the data portal *CKAN*[7]. This portal is being extended to allow SPARQL queries over the repository. With that, we close the whole Linked Open Data cycle. Data is accessed and transformed into RDF using extraction and storage components, then it is augmented and interlinked with other data sources (found through online data portals) and finally the newly created dataset is published as a new datasource on the web, announcing itself to the world via a data portal and ready to be used. Both, announcement as well as discovery via CKAN is an integral part of the LOD2 Stack.

4 Use-Case: Facilitating Data-Flows at a Global Publisher

Wolters Kluwer is a global knowledge and information service provider with more than 19.000 employees worldwide and core competencies in the legal, tax and business domains. Wolters Kluwer offers information for the professional in any format including folio, software and services.

The Linked Data life-cycle mirrored to publishing business. The steps described in the life-cycle highly resemble traditional workflow steps in a publishing house. Therefore, conceptually adopting this life-cycle for the publishing business is very reasonable. In traditional publishing, the focus is mainly on textual information, starting from the authoring process up to layout and printing. Metadata has recently become prominent with increasing use of digital libraries with sophisticated search functionalities. This shift of scope is still ongoing, and new company internal processes and skills must be developed and implemented. Since the LOD2 Stack tools are, by definition, (meta-)data oriented and highly standard compliant, they have great potential to fill the gap between very efficient content processing and very flexible and powerful metadata management. As a first step, we have focused on the following parts of the life-cycle:

Extraction: Usually, the content in a publisher's house is stored in XML, and stored in the same file as the text. Therefore, the extraction of the metadata is an important step in the overall process.

Storage: This is the foundation for everything. All metadata must be accessible to all tools exploiting it.

Authoring: Human editors must be able to code their knowledge domain in an easy way, which also means that features for proper maintenance and development must be in place.

Interlinking: When the publishing industry talks about "linking", it is mainly referring to hyperlinks in text. The capabilities here are different, meaning linking different knowledge sources in order to create a semantic network.

Search/Browse/Exploration: A proper UI allowing editorial staff to interact with the data is key in an operational environment. The gap between technological representation and semantic human interpretation must be bridged by

[7] `http://ckan.net`

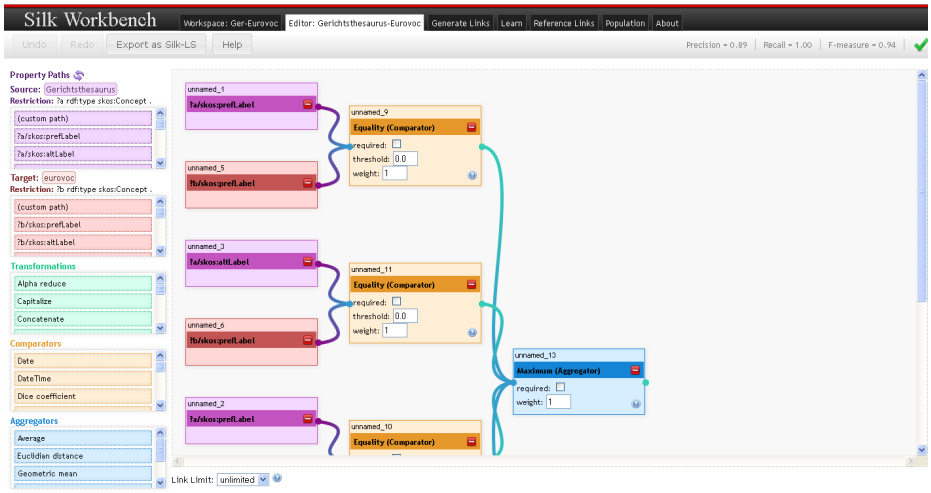

Fig. 5. Silk workbench with loaded linking specification to link law courts from different datasets.

using metaphors and proper on-the-fly mapping between URIs and human-readable labels.

Based on these core tasks, tools from the LOD2 Stack were selected in order to fulfill the respective requirements. This resulted in a working prototype called *Pebbles*, using and integrating the following tools:

- Virtuoso triple store for storage of the triples, along with its WebDAV environment for storage of the accompanying XML source files.
- PoolParty for maintaining all the controlled vocabularies, including domain taxonomies and thesauri. This environment is also used for publishing Linked Data. Initially, labour law thesaurus and a court thesaurus have been made publicly available under a Creative Commons license.
- SILK framework for mapping between the Wolters Kluwer knowledge bases and external sources like DBpedia or the EUROVOC thesaurus.
- OntoWiki as the user interface for human end users. Features for taxonomy browsing and filtering, but also for metadata management like adding or deleting or changing an instance, are used. There is also a connector to the original XML file, so that the basic text information can be displayed in parallel, rendered in HTML.
- Valiant and VEnrich (a wrapper around PoolParty Extractor) tools to make the extraction process efficient and performant.

The resulting prototype (in fact an LOD2 Stack adoption) is currently being evaluated by the operational editorial team, to assess its appropriateness as a basis for an internal knowledge base within Wolters Kluwer Germany.

Our evaluation included, in particular, a dataset extracted from approximately 800.000 semi-structured XML documents from the German legal domain. From these documents, 46.651.884 facts have been extracted. This process was

run in batch mode on a server with 8 GB memory and takes approximately 4 hours. The data is strongly linked within itself (the documents refer to other documents in the document set). The exploration of linking to external public sources has started. One of the few German sources available is the German DBpedia. Using Silk, we were able to discover links to all German laws.

Opportunities beyond local business. The technology at hand has three main characteristics, which make it a candidate for usage in a global environment: it is about semantics, it is about connecting these semantics and it is about referring to official international standards. Imagine a global publisher with businesses in more than 40 countries worldwide. In order to offer cross-country offerings in different languages, there are three approaches possible:

- Approaching each and every country individually and collecting the data on an individual basis.
- Introducing a global content repository.
- Introducing a semantic layer on top of every local repository for automatic extraction and bundling of data.

The first approach needs many effective and controlled workflows in place, in order to be effective and efficient. The second approach is very expensive and time consuming to implement. The third approach seems to be the best compromise and most sustainable solution and is thus favored at Wolters Kluwer.

Summary and next steps. The LOD2 Stack serves the needs of a publishing use case in many respects: The LOD life cycle reflects very well the tasks a publishing house has to perform; getting a grip on semantics will be a key skill of professionals and therefore also of their service providers; the wide usage of standards ensures the flexibility of not being locked in to a specific tool or vendor. The Pebbles prototype has shown that some of the tools in the stack are mature enough, so that they can be used in an industrial environment.

Currently, we are looking at expanding the usage of the LOD2 Stack in our use case, mainly by including NLP tools in order to address the classification/enrichment step of the life-cycle. If this is successful, a lot of additional added-value to our data and therefore to our products can automatically be exploited. In addition, we want to use the publishing capabilities of *PoolParty* in order to publish our data as LOD data and therefore get in touch with the developer community. We seek a win-win situation, where our data is more widely used and requirements for additional or completely new data can be met by us.

5 Conclusion and Outlook

In this article we presented the LOD2 Stack, the result of a large-scale effort to provide technological support for the life-cycle of Linked Data. We deem this a first step in a larger research and development agenda, where derivatives of the LOD2 Stack are employed to create corporate enterprise knowledge hubs withing the Intranets of large companies such as the publisher Wolters Kluwer. In

order to realize our vision, we aim to further strengthen the light-weight REST-API based integration between the components of the stack. The overall stack architecture and guidelines can also serve as a blue-print for similar software stacks in other areas. For the next iterations of the LOD2 Stack, we plan to increase tool coverage and to include more 3rd party developed tools.

References

1. S. Auer, S. Dietzold, and T. Riechert. OntoWiki - A Tool for Social, Semantic Collaboration. In *5th Int. Semantic Web Conference, ISWC 2006*, volume 4273 of *LNCS*, pages 736–749. Springer, 2006.
2. C. Bizer. D2r map - a database to rdf mapping language. In *WWW (Posters)*, 2003.
3. F. Christ and B. Nagel. A reference architecture for semantic content management systems. In *4th Int. Workshop on Enterprise Modelling and Information Systems Architectures, EMISA 2011*, volume 190 of *LNI*, pages 135–148. GI, 2011.
4. O. Erling. Virtuoso, a hybrid rdbms/graph column store. *IEEE Data Eng. Bull.*, 35(1):3–8, 2012.
5. A. Jentzsch, R. Isele, and C. Bizer. Silk - generating rdf links while publishing or consuming linked data. In *ISWC 2010 Posters & Demo Track*, volume 658. CEUR-WS.org, 2010.
6. J. Lehmann. DL-Learner: learning concepts in description logics. *Journal of Machine Learning Research (JMLR)*, 10:2639–2642, 2009.
7. J. Lehmann, S. Auer, L. Bühmann, and S. Tramp. Class expression learning for ontology engineering. *Journal of Web Semantics*, 9:71 – 81, 2011.
8. J. Lehmann and L. Bühmann. Ore - a tool for repairing and enriching knowledge bases. In *9th Int. Semantic Web Conference (ISWC2010)*, LNCS. Springer, 2010.
9. J. Lehmann and P. Hitzler. Concept learning in description logics using refinement operators. *Machine Learning journal*, 78(1-2):203–250, 2010.
10. P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. Dbpedia spotlight: Shedding light on the web of documents. In *7th I-Semantics*, 2011.
11. P. N. Mendes, H. Mühleisen, and C. Bizer. Sieve: Linked Data Quality Assessment and Fusion. In *2nd Int. WS on Linked Web Data Mgmt (LWDM 2012) at EDBT 2012*, 2012.
12. I. Murdock. The Debian Manifesto. http://www.debian.org/doc/manuals/project-history/ap-manifesto.en.html, 1994.
13. A.-C. Ngonga Ngomo and S. Auer. Limes - a time-efficient approach for large-scale link discovery on the web of data. In *IJCAI*, 2011.
14. T. Schandl and A. Blumauer. Poolparty: Skos thesaurus management utilizing linked data. In *7th Extended Semantic Web Conf., ESWC 2010*, volume 6089 of *LNCS*, pages 421–425. Springer, 2010.
15. S. Tramp, P. Frischmuth, T. Ermilov, and S. Auer. Weaving a Social Data Web with Semantic Pingback. In *EKAW 2010*, volume 6317 of *LNAI*, pages 135–149. Springer.
16. G. Tummarello, R. Cyganiak, M. Catasta, S. Danielczyk, R. Delbru, and S. Decker. Sig.ma: Live views on the web of data. *J. Web Sem.*, 8(4):355–364, 2010.