Knowledge Base Creation, Enrichment and Repair

Sebastian Hellmann[1](✉), Volha Bryl[2], Lorenz Bühmann[1], Milan Dojchinovski[4], Dimitris Kontokostas[1], Jens Lehmann[1], Uroš Milošević[3], Petar Petrovski[2], Vojtěch Svátek[4], Mladen Stanojević[3], and Ondřej Zamazal[4]

[1] University of Leipzig, Leipzig, Germany
{hellmann,buehmann,kontokostas,lehmann}@informatik.uni-leipzig.de
[2] University of Mannheim, Mannheim, Germany
{volha,petar}@informatik.uni-mannheim.de
[3] Institute Mihajlo Pupin, Belgrade, Serbia
{uros.milosevic,mladen.stanojevic}@pupin.rs
[4] University of Economics Prague, Prague, Czech Republic
{milan.dojchinovski,svatek,ondrej.zamazal}@vse.cz

Abstract. This chapter focuses on data transformation to RDF and Linked Data and furthermore on the improvement of existing or extracted data especially with respect to schema enrichment and ontology repair. Tasks concerning the triplification of data are mainly grounded on existing and well-proven techniques and were refined during the lifetime of the LOD2 project and integrated into the LOD2 Stack. Triplification of legacy data, i.e. data not yet in RDF, represents the entry point for legacy systems to participate in the LOD cloud. While existing systems are often very useful and successful, there are notable differences between the ways knowledge bases and Wikis or databases are created and used. One of the key differences in content is in the importance and use of schematic information in knowledge bases. This information is usually absent in the source system and therefore also in many LOD knowledge bases. However, schema information is needed for consistency checking and finding modelling problems. We will present a combination of enrichment and repair steps to tackle this problem based on previous research in machine learning and knowledge representation. Overall, the Chapter describes how to enable tool-supported creation and publishing of RDF as Linked Data (Sect. 1) and how to increase the quality and value of such large knowledge bases when published on the Web (Sect. 2).

1 Linked Data Creation and Extraction

1.1 DBpedia, a Large-Scale, Multilingual Knowledge Base Extracted from Wikipedia

Wikipedia is the 6th most popular website[1], the most widely used encyclopedia, and one of the finest examples of truly collaboratively created content. There are

[1] http://www.alexa.com/topsites. Retrieved in May 2014.

S. Auer et al. (Eds.): Linked Open Data, LNCS 8661, pp. 45–69, 2014.
DOI: 10.1007/978-3-319-09846-3_3

official Wikipedia editions in 287 different languages which range in size from a couple of hundred articles up to 3.8 million articles (English edition)[2]. Besides of free text, Wikipedia articles consist of different types of structured data such as infoboxes, tables, lists, and categorization data. Wikipedia currently offers only free-text search capabilities to its users. Using Wikipedia search, it is thus very difficult to find all rivers that flow into the Rhine and are longer than 100 km, or all Italian composers that were born in the 18th century.

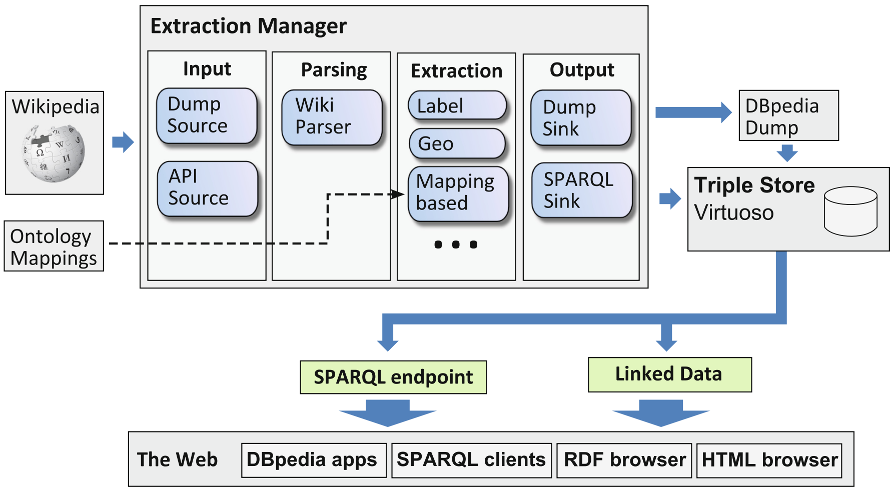


Fig. 1. Overview of DBpedia extraction framework

The DBpedia project [9,13,14] builds a large-scale, multilingual knowledge base by extracting structured data from Wikipedia editions in 111 languages. Wikipedia editions are extracted by the open source "DBpedia extraction framework" (cf. Fig. 1). The largest DBpedia knowledge base which is extracted from the English edition of Wikipedia consists of over 400 million facts that describe 3.7 million things. The DBpedia knowledge bases that are extracted from the other 110 Wikipedia editions together consist of 1.46 billion facts and describe 10 million additional things. The extracted knowledge is encapsulated in modular dumps as depicted in Fig. 2. This knowledge base can be used to answer expressive queries such as the ones outlined above. Being multilingual and covering an wide range of topics, the DBpedia knowledge base is also useful within further application domains such as data integration, named entity recognition, topic detection, and document ranking.

The DBpedia knowledge base is widely used as a test-bed in the research community and numerous applications, algorithms and tools have been built around or applied to DBpedia. Due to the continuous growth of Wikipedia and

[2] http://meta.wikimedia.org/wiki/List_of_Wikipedias

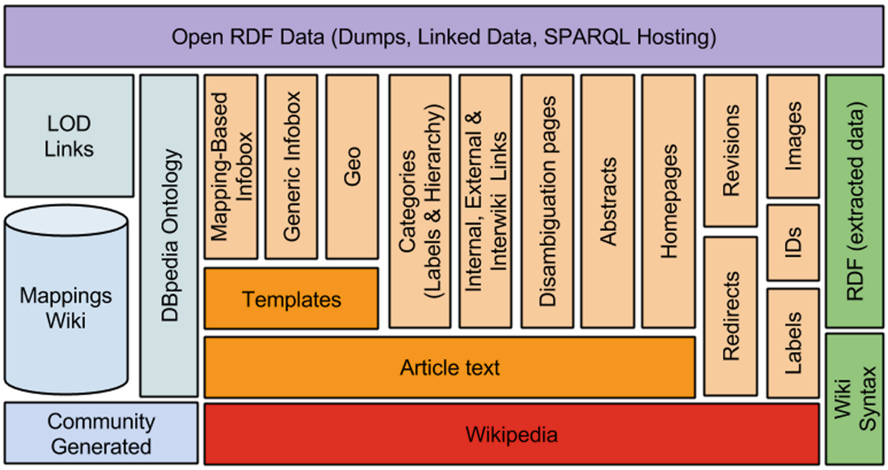


Fig. 2. Overview of the DBpedia data stack.

improvements in DBpedia, the extracted data provides an increasing added value for data acquisition, re-use and integration tasks within organisations. While the quality of extracted data is unlikely to reach the quality of completely manually curated data sources, it can be applied to some enterprise information integration use cases and has shown to be relevant in several applications beyond research projects. DBpedia is served as Linked Data on the Web. Since it covers a wide variety of topics and sets RDF links pointing into various external data sources, many Linked Data publishers have decided to set RDF links pointing to DBpedia from their data sets. Thus, DBpedia became a central interlinking hub in the Web of Linked Data and has been a key factor for the success of the Linked Open Data initiative.

The structure of the DBpedia knowledge base is maintained by the DBpedia user community. Most importantly, the community creates mappings from Wikipedia information representation structures to the DBpedia ontology. This ontology unifies different template structures, both within single Wikipedia language editions and across currently 27 different languages. The maintenance of different language editions of DBpedia is spread across a number of organisations. Each organisation is responsible for the support of a certain language. The local DBpedia chapters are coordinated by the DBpedia Internationalisation Committee. The *DBpedia Association* provides an umbrella on top of all the DBpedia chapters and tries to support DBpedia and the DBpedia Contributors Community.

1.2 RDFa, Microdata and Microformats Extraction Framework

In order to support web applications to understand the content of HTML pages, an increasing number of websites have started to semantically markup their pages, that is, embed structured data describing products, people, organizations, places, events, etc. into HTML pages using such markup standards as Microformats[3], RDFa[4] and Microdata[5]. Microformats use style definitions to annotate HTML text with terms from a fixed set of vocabularies, RDFa allows embedding any kind of RDF data into HTML pages, and Microdata is part of the HTML5 standardization effort allowing the use of arbitrary vocabularies for structured data.

The embedded data is crawled together with the HTML pages by search engines, such as Google, Yahoo! and Bing, which use these data to enrich their search results. Up to now, only these companies were capable of providing insights [15] into the amount as well as the types of data that are published on the web using different markup standards as they were the only ones possessing large-scale web crawls. However, the situation changed with the advent of the Common Crawl[6], a non-profit foundation that crawls the web and regularly publishes the resulting corpora for public usage on Amazon S3.

For the purpose of extracting structured data from these large-scale web corpora we have developed the *RDFa, Microdata and Microformats extraction framework* that is available online[7].

The extraction consists of the following steps. Firstly, a file with the crawled data, in the form of ARC or WARC archive, is *downloaded* from the storage. The archives usually contain up to several thousands of archived web pages. The framework relies on the Anything To Triples (Any23)[8] parser library for *extracting* RDFa, Microdata, and Microformats from HTML content. Any23 outputs RDF quads, consisting of subject, predicate, object, and a URL which identifies the HTML page from which the triple was extracted. Any23 parses web pages for structured data by building a DOM tree and then evaluates XPath expressions to extract the structured data. As we have found that the tree generation accounts for much of the parsing cost, we have introduced the *filtering* step: We run regular expressions against each archived HTML page prior to extraction to detect the presence of structured data, and only run the Any23 extractor when potential matches are found. The *output* of the extraction process is in NQ (RDF quads) format.

We have made available two implementations of the extraction framework, one based on the Amazon Web Services, and the second one being a Map/Reduce implementation that can be run over any Hadoop cluster. Additionally, we provide a plugin to the Apache Nutch crawler allowing the user to configure the crawl and then extract structured data from the resulting page corpus.

To verify the framework, three large scale RDFa, Microformats and Microdata extractions have been performed, corresponding to the Common Crawl

[3] http://microformats.org/
[4] http://www.w3.org/TR/xhtml-rdfa-primer/
[5] http://www.w3.org/TR/microdata/
[6] http://commoncrawl.org/
[7] https://subversion.assembla.com/svn/commondata/
[8] https://any23.apache.org/

data from 2009/2010, August 2012 and November 2013. The results of the 2012 and 2009/2010 are published in [2] and [16], respectively. Table 1 presents the comparative summary of the three extracted datasets. The table reports the number and the percentage of URLs in each crawl containing structured data, and gives the percentage of these data represented using Microformats, RDFa and Microdata, respectively.

Table 1. Large-scale RDF datasets extracted from Common Crawl (CC): summary

	CC 2009/2010	CC August 2012	CC November 2013
Size(TB), compressed	28.9	40.1	44
Size, URLs	2,565,741,671	3,005,629,093	2,224,829,946
Size, Domains	19,113,929	40,600,000	12,831,509
Parsing cost, USD	576	398	263
Structured data, URLs with triples	147,871,837	369,254,196	585,792,337
Structured data, in %	5.76	12.28	26.32
Microformats, in %	96.99	70.98	47.48
RDFa, in %	2.81	22.71	26.48
Microdata, in %	0.2	6.31	26.04
Average num. of triples per URL	3.35	4.05	4.04

The numbers illustrate the trends very clearly: in the recent years, the amount of structured data embedded into HTML pages keeps increasing. The use of Microformats is decreasing rapidly, while the use of RDFa and especially Microdata standards has increased a lot, which is not surprising as the adoption of the latter is strongly encouraged by the biggest search engines. On the other hand, the average number of triples per web page (only pages containing structured data are considered) stays the same through the different version of the crawl, which means that the data completeness has not changed much.

Concerning the topical domains of the published data, the dominant ones are: persons and organizations (for all three formats), blog- and CMS-related metadata (RDFa and Microdata), navigational metadata (RDFa and Microdata), product data (all three formats), and event data (Microformats). Additional topical domains with smaller adoption include job postings (Microdata) and recipes (Microformats). The data types, formats and vocabularies seem to be largely determined by the major consumers the data is targeted at. For instance, the RDFa portion of the corpora is dominated by the vocabulary promoted by Facebook, while the Microdata subset is dominated by the vocabularies promoted by Google, Yahoo! and Bing via schema.org.

More detailed statistics on the three corpora are available at the Web Data Commons page[9].

By publishing the data extracted from RDFa, Microdata and Microformats annotations, we hope on the one hand to initialize further domain-specific studies by third parties. On the other hand, we hope to lay the foundation for enlarging the number of applications that consume structured data from the web.

1.3 Rozeta

The ever-growing world of data is largely unstructured. It is estimated that information sources such as books, journals, documents, social media content and everyday news articles constitute as much as 90 % of it. Making sense of all this data and exposing the knowledge hidden beneath, while minimizing human effort, is a challenging task which often holds the key to new insights that can prove crucial to one's research or business. Still, understanding the context, and finding related information are hurdles that language technologies are yet to overcome.

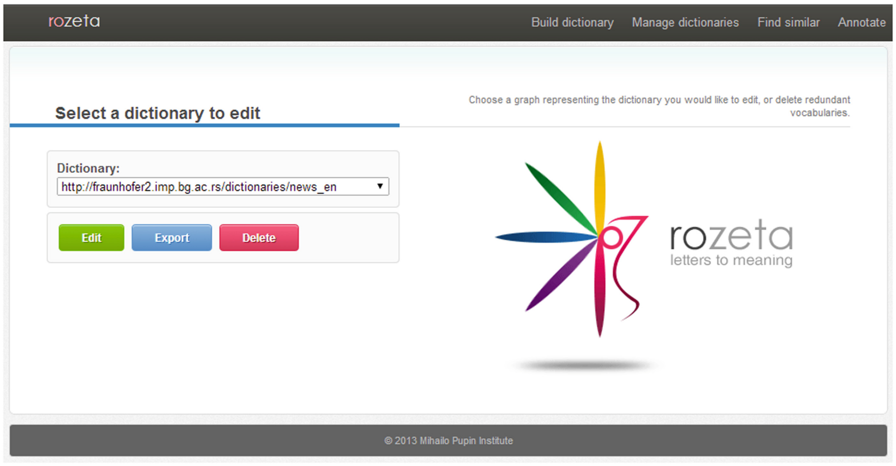
Rozeta is a multilingual NLP and Linked Data tool wrapped around STRUTEX, a structured text knowledge representation technique, used to extract words and phrases from natural language documents and represent them in a structured form. Originally designed for the needs of Wolters Kluwer Deutschland, for the purposes of organizing and searching through their database of court cases (based on numerous criteria, including case similarity), Rozeta provides automatic extraction of STRUTEX dictionaries in Linked Data form, semantic enrichment through link discovery services, a manual revision and authoring component, a document similarity search tool and an automatic document classifier (Fig. 3).

1.3.1 Dictionary Management

The Rozeta dictionary editor (Fig. 4) allows for a quick overview of all dictionary entries, as well as semi-automatic (supervised) vocabulary enrichment/link discovery and manual cleanup. It provides a quick-filter/AJAX search box that helps users swiftly browse through the dictionary by retrieving the entries that start with a given string, on-the-fly. The detailed view for a single entry shows its URI, text, class, any existing links to relevant LOD resources, as well as links to the files the entry originated from. Both the class and file origin information can be used as filters, which can help focus one's editing efforts on a single class or file, respectively.

To aid the user in enriching individual entries with links to other relevant linked data sources, Wiktionary2RDF recommendations are retrieved automatically. The user can opt for one of the available properties (`skos:exactMatch` and `skos:relatedMatch`) or generate a link using a custom one. Furthermore, the *Custom link* and *More links* buttons give the user the ability to link the

[9] http://webdatacommons.org



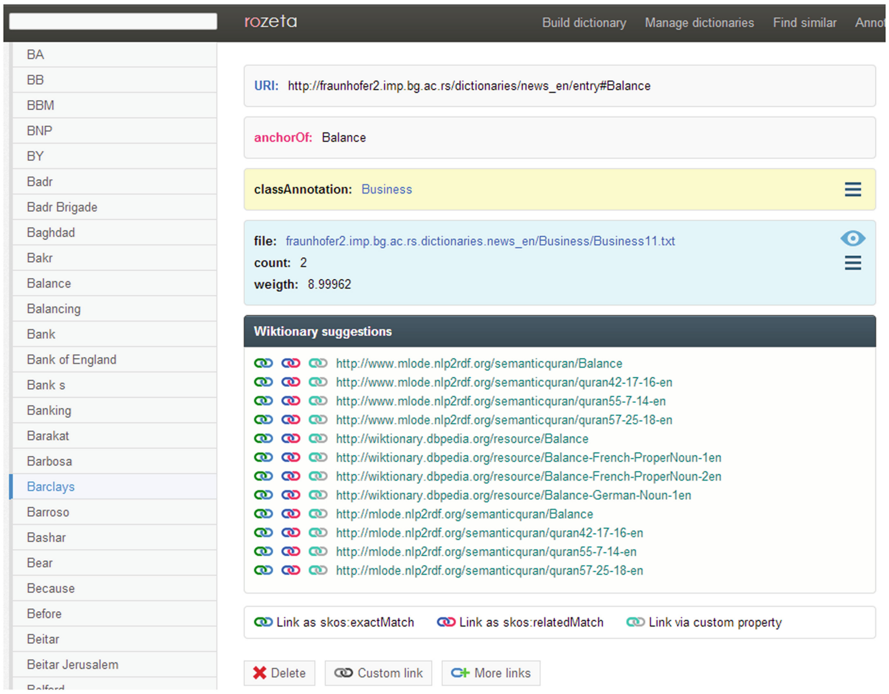
Select a dictionary to edit

Choose a graph representing the dictionary you would like to edit, or delete redundant vocabularies.

Dictionary: http://fraunhofer2.imp.bg.ac.rs/dictionaries/news_en

© 2013 Mihailo Pupin Institute

Fig. 3. Rozeta: dictionary selection



URI: http://fraunhofer2.imp.bg.ac.rs/dictionaries/news_en/entry#Balance

anchorOf: Balance

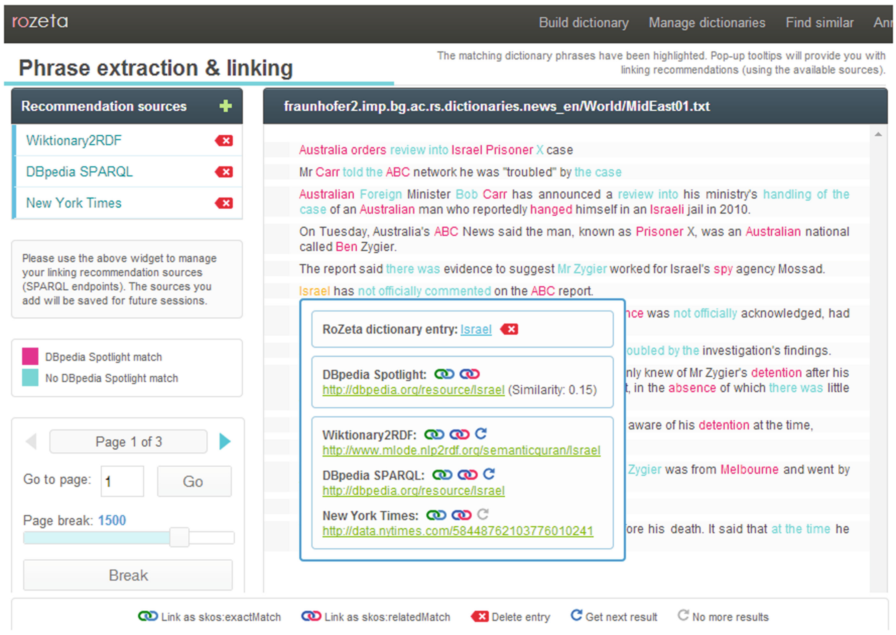
classAnnotation: Business

file: fraunhofer2.imp.bg.ac.rs.dictionaries.news_en/Business/Business11.txt
count: 2
weigth: 8.99962

http://www.mlode.nlp2rdf.org/semanticquran/Balance
http://www.mlode.nlp2rdf.org/semanticquran/quran42-17-16-en
http://www.mlode.nlp2rdf.org/semanticquran/quran55-7-14-en
http://www.mlode.nlp2rdf.org/semanticquran/quran57-25-18-en
http://wiktionary.dbpedia.org/resource/Balance
http://wiktionary.dbpedia.org/resource/Balance-French-ProperNoun-1en
http://wiktionary.dbpedia.org/resource/Balance-French-ProperNoun-2en
http://wiktionary.dbpedia.org/resource/Balance-German-Noun-1en
http://mlode.nlp2rdf.org/semanticquran/Balance
http://mlode.nlp2rdf.org/semanticquran/quran42-17-16-en
http://mlode.nlp2rdf.org/semanticquran/quran55-7-14-en
http://mlode.nlp2rdf.org/semanticquran/quran57-25-18-en

Fig. 4. Rozeta: dictionary management

selected dictionary phrase to any LOD resource, either manually, or by letting the system provide them with automatic recommendations through one of the available link discovery services, such as Sindice or a custom SPARQL endpoint.



The matching dictionary phrases have been highlighted. Pop-up tooltips will provide you with linking recommendations (using the available sources).

Recommendation sources
- Wiktionary2RDF
- DBpedia SPARQL
- New York Times

Please use the above widget to manage your linking recommendation sources (SPARQL endpoints). The sources you add will be saved for future sessions.

🟥 DBpedia Spotlight match
🟦 No DBpedia Spotlight match

Page 1 of 3
Go to page: 1 Go
Page break: 1500
Break

fraunhofer2.imp.bg.ac.rs.dictionaries.news_en/World/MidEast01.txt

Australia orders review into Israel Prisoner X case
Mr Carr told the ABC network he was "troubled" by the case
Australian Foreign Minister Bob Carr has announced a review into his ministry's handling of the case of an Australian man who reportedly hanged himself in an Israeli jail in 2010.
On Tuesday, Australia's ABC News said the man, known as Prisoner X, was an Australian national called Ben Zygier.
The report said there was evidence to suggest Mr Zygier worked for Israel's spy agency Mossad.
Israel has not officially commented on the ABC report.
nce was not officially acknowledged, had
oubled by the investigation's findings.
nly knew of Mr Zygier's detention after his
, in the absence of which there was little
aware of his detention at the time,
Zygier was from Melbourne and went by
ore his death. It said that at the time he

RoZeta dictionary entry: Israel
DBpedia Spotlight: http://dbpedia.org/resource/Israel (Similarity: 0.15)
Wiktionary2RDF: http://www.mlode.nlp2rdf.org/semanticquran/Israel
DBpedia SPARQL: http://dbpedia.org/resource/Israel
New York Times: http://data.nytimes.com/58448762103776010241

Link as skos:exactMatch | Link as skos:relatedMatch | Delete entry | Get next result | No more results

Fig. 5. Rozeta: text annotation and enrichment

1.3.2 Text Annotation and Enrichment

The text annotation and enrichment module, used for highlighting the learned vocabulary entries in any natural language document and proposing potential links through custom services, can be launched from the dictionary editor, or used as a stand-alone application.

The highlighted words and phrases hold links to the corresponding dictionary entry pages, as well as linking recommendations from DBpedia Spotlight, or custom SPARQL endpoints (retrieved on-the-fly; sources are easily managed through an accompanying widget). The pop-up widget also generates quick-link buttons (skos:exactMatch and skos:relatedMatch) for linking the related entries to recommended Linked Open Data resources (Fig. 5).

2 Analysis, Enrichment and Repair of Linked Data with ORE Tool

The ORE tool supports knowledge engineers in enriching the schema of OWL based knowledge bases, either accessible as file or via SPARQL. Additionally, it allows for the detection and repair of logical errors as well as the validation of instance data by defining constraints in forms of OWL axioms. ORE also integrates the PaOMat framework (see Sect. 3), thus, it allows for the detection and repair of naming issues. The ORE tool is published as an open source project[10,11].

2.1 Logical Debugging

2.1.1 Motivation

Along with the uptake of Semantic Web technologies, we observe a steady increase of the amount of available OWL ontologies as well as an increase of the complexity of such ontologies. While the expressiveness of OWL is indeed a strong feature, it can also lead to a *misunderstanding* and *misusage* of particular types of constructs in the language. In turn, this can lead to modeling errors in the ontology, i.e. *inconsistency* or *unsatisfiable classes.* Inconsistency, in simple terms, is a logical contradiction in the knowledge base, which makes it impossible to derive any meaningful information by applying standard OWL reasoning techniques. Unsatisfiable classes usually are a fundamental modeling error, in that they cannot be used to characterize any individual, that means they cannot have any individual.

Both kinds of modeling errors are quite easy to detect by standard OWL reasoners, however, determining why the errors hold can be a considerable challenge even for experts in the formalism and in the domain, even for modestly sized ontologies. The problem worsens significantly as the number and complexity of axioms of the ontology grows. Clearly, only with the understanding of why such an undesired entailment holds, it is possible to get rid of the errors, i.e. to *repair* the ontology.

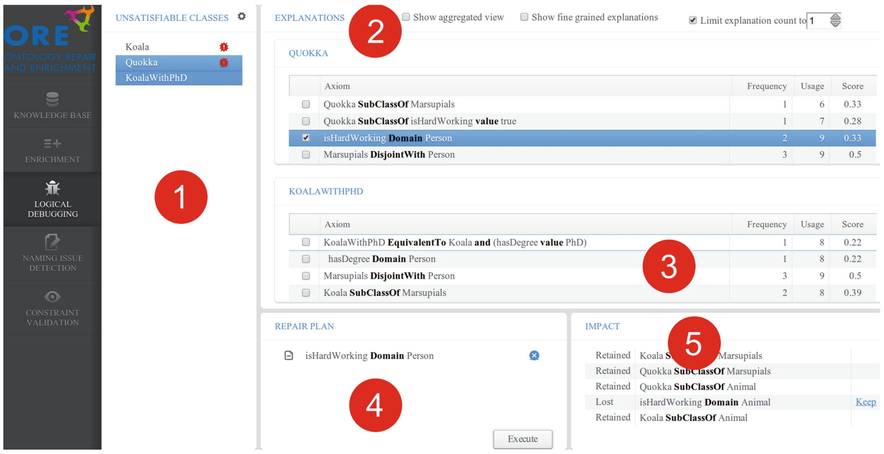
In the area of *ontology debugging*, a specific type of explanation called *justifications* [1,7,8,18] was introduced by the research community, which is basically a minimal subset of the ontology that is sufficient for the entailment to hold. The set of axioms corresponding to the justification is minimal in the sense that if an axiom is removed from the set, the remaining axioms no longer support the entailment. One such justification could be the following example, which gives an explanation why the class `metal` is unsatisfiable.

```
metal EquivalentTo chemical and (atomic-number some integer)
                            and (atomic-number exactly 1 Thing)
nonmetal EquivalentTo chemical and (atomic-number some integer)
                               and (atomic-number exactly 1 Thing)
metal DisjointWith nonmetal
```

2.1.2 Support in ORE

The debugging view for OWL ontologies (see Fig. 6), here described for unsatisfiable classes, consists mainly of four parts: The first part on the left side (1) gives

[10] http://ore-tool.net/Projects/ORE
[11] https://github.com/AKSW/ORE



QUOKKA

	Axiom	Frequency	Usage	Score
☐	Quokka **SubClassOf** Marsupials	1	6	0.33
☐	Quokka **SubClassOf** isHardWorking **value** true	1	7	0.28
✓	isHardWorking **Domain** Person	2	9	0.33
☐	Marsupials **DisjointWith** Person	3	9	0.5

KOALAWITHPHD

	Axiom	Frequency	Usage	Score
☐	KoalaWithPhD **EquivalentTo** Koala **and** (hasDegree **value** PhD)	1	8	0.22
☐	hasDegree **Domain** Person	1	8	0.22
☐	Marsupials **DisjointWith** Person	3	9	0.5
☐	Koala **SubClassOf** Marsupials	2	8	0.39

REPAIR PLAN

isHardWorking **Domain** Person

IMPACT

Retained	Koala **S** [illegible] Marsupials	
Retained	Quokka **SubClassOf** Marsupials	
Retained	Quokka **SubClassOf** Animal	
Lost	isHardWorking **Domain** Animal	Keep
Retained	Koala **SubClassOf** Animal	

Fig. 6. Screenshot of the debugging view for OWL ontologies.

a list of the unsatisfiable classes which were detected in the selected knowledge base. In this list itself, root unsatisfiable classes, i.e. classes which unsatisfiability does not depend on the unsatisfiability of other classes, are tagged with "Root". Usually, in a repair process it is recommended to handle such root unsatisfiable classes first, as other conflicts will be solved then too. The second main part contains the presentation of the explanations, which are computed once an unsatisfiable class is selected, and shown as tables (3). In addition to the axioms of the justification, each of the tables contains two metrics which give some insights into how an axiom is involved in other justifications (frequency) and how strong an axiom is connected in the ontology (usage), both metrics finally aggregated in total score (score). The menu (2) allows for the choice between the computation of regular or laconic justifications as well as for the limitation of the maximum number of computed justifications. Furthermore, it gives the option to show an aggregated view of all the axioms contained in the computed justifications, compared to the presentation of each justification in its own table. In the third part (4) - the repair plan - a list of all changes a user has chosen in order to repair the knowledge base is displayed. The changes can either be the removal or addition of axioms and will be executed once a user has decided to do so by clicking the "Execute" button. The last part of the debugging view, located at the bottom right (5), contains an outline of the effects the changes of the repair plan would have to the knowledge base, i.e. it contains lost and retained entailments. If an entailment is found to be lost when executing the repair plan, it is possible to add an axiom to the knowledge base which retains that entailment. This part is only available during the debugging of unsatisfiable classes, as it is (currently) impossible to compute such entailments in inconsistent knowledge bases.

2.2 Schema Enrichment

2.2.1 Motivation

The Semantic Web has recently seen a rise in the availability and usage of knowledge bases, as can be observed within the Linking Open Data Initiative, the TONES and Protégé ontology repositories, or the Watson search engine. Despite this growth, there is still a lack of knowledge bases that consist of sophisticated schema information and instance data adhering to this schema. Several knowledge bases, e.g. in the life sciences, only consist of schema information, while others are, to a large extent, a collection of facts without a clear structure, e.g. information extracted from data bases or texts. The combination of sophisticated schema and instance data would allow powerful reasoning, consistency checking, and improved querying possibilities. Schema enrichment allows to create a sophisticated schema base based on existing data (sometimes referred to as "grass roots" approach or "after the fact" schema creation).

Example 1. As an example, consider a knowledge base containing a class `Capital` and instances of this class, e.g. London, Paris, Washington, Canberra, etc. A machine learning algorithm could, then, suggest that the class `Capital` may be equivalent to one of the following OWL class expressions in Manchester OWL syntax[12]:

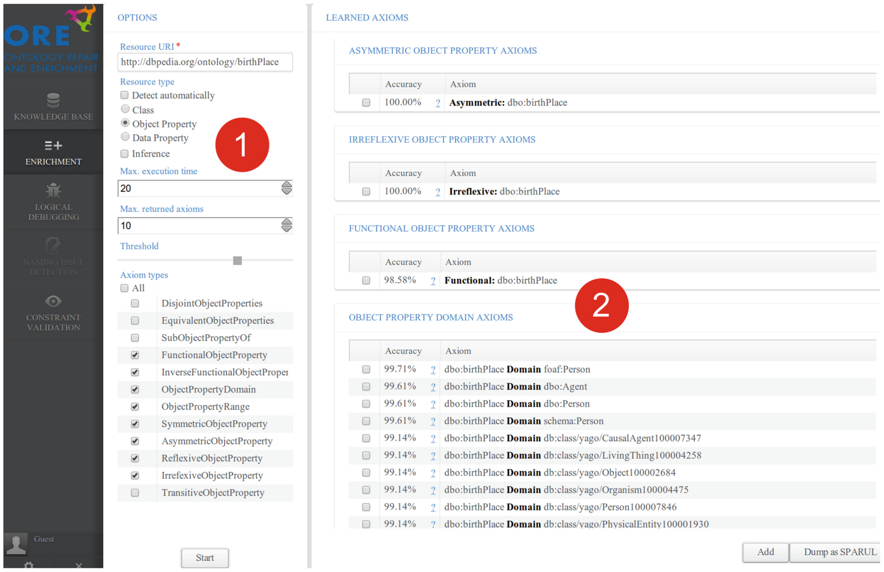
```
City and isCapitalOf min 1 GeopoliticalRegion
City and isCapitalOf min 1 Country
```

Both suggestions could be plausible: The first one is more general and includes cities that are capitals of states, whereas the latter one is stricter and limits the instances to capitals of countries. A knowledge engineer can decide which one is more appropriate, i.e. a semi-automatic approach is used, and the machine learning algorithm should guide the user by pointing out which one fits the existing instances better.

Assuming the knowledge engineer decides for the latter, an algorithm can show the user whether there are instances of the class `Capital` which are neither instances of `City` nor related via the property `isCapitalOf` to an instance of `Country`.[13] The knowledge engineer can then continue to look at those instances and assign them to a different class as well as provide more complete information; thus improving the quality of the knowledge base. After adding the definition of `Capital`, an OWL reasoner can compute further instances of the class which have not been explicitly assigned before.

[12] For details on Manchester OWL syntax (e.g. used in Protégé, OntoWiki) see http://www.w3.org/TR/owl2-manchester-syntax/.

[13] This is not an inconsistency under the standard OWL open world assumption, but rather a hint towards a potential modelling error.



OPTIONS

Resource URI *
http://dbpedia.org/ontology/birthPlace

Resource type
- Detect automatically
- Class
- [✓ Object Property]
- Data Property
- Inference

Max. execution time: 20

Max. returned axioms: 10

Threshold

Axiom types
- All
- DisjointObjectProperties
- EquivalentObjectProperties
- SubObjectPropertyOf
- [✓ FunctionalObjectProperty]
- [✓ InverseFunctionalObjectProper]
- [✓ ObjectPropertyDomain]
- [✓ ObjectPropertyRange]
- [✓ SymmetricObjectProperty]
- [✓ AsymmetricObjectProperty]
- [✓ ReflexiveObjectProperty]
- [✓ IrrefexiveObjectProperty]
- TransitiveObjectProperty

LEARNED AXIOMS

ASYMMETRIC OBJECT PROPERTY AXIOMS

Accuracy		Axiom
100.00%	?	**Asymmetric:** dbo:birthPlace

IRREFLEXIVE OBJECT PROPERTY AXIOMS

Accuracy		Axiom
100.00%	?	**Irreflexive:** dbo:birthPlace

FUNCTIONAL OBJECT PROPERTY AXIOMS

Accuracy		Axiom
98.58%	?	**Functional:** dbo:birthPlace

OBJECT PROPERTY DOMAIN AXIOMS

Accuracy		Axiom
99.71%	?	dbo:birthPlace **Domain** foaf:Person
99.61%	?	dbo:birthPlace **Domain** dbo:Agent
99.61%	?	dbo:birthPlace **Domain** dbo:Person
99.61%	?	dbo:birthPlace **Domain** schema:Person
99.14%	?	dbo:birthPlace **Domain** db:class/yago/CausalAgent100007347
99.14%	?	dbo:birthPlace **Domain** db:class/yago/LivingThing100004258
99.14%	?	dbo:birthPlace **Domain** db:class/yago/Object100002684
99.14%	?	dbo:birthPlace **Domain** db:class/yago/Organism100004475
99.14%	?	dbo:birthPlace **Domain** db:class/yago/Person100007846
99.14%	?	dbo:birthPlace **Domain** db:class/yago/PhysicalEntity100001930

Fig. 7. Screenshot of the enrichment view for SPARQL knowledge bases.

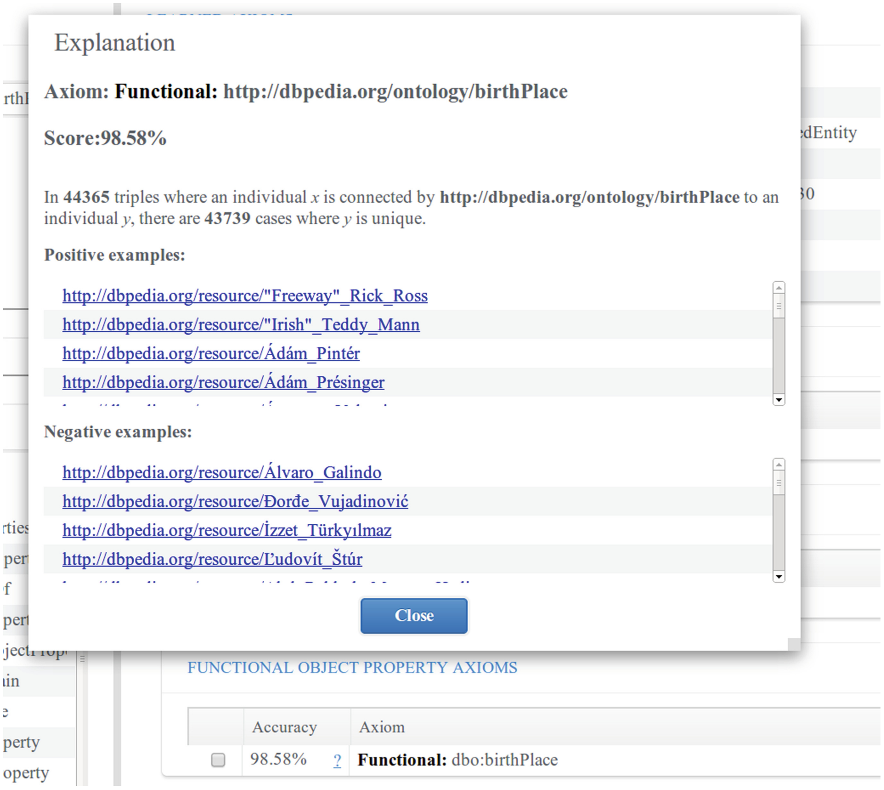
2.2.2 Support in ORE

The enrichment view for SPARQL knowledge bases(see Fig. 7), can be subdivided into two main parts: The first part on the left side (①) allows for configuring the enrichment process like to denote for which entity and which types ORE will search for schema axioms. The second part on the right side(②) shows the generated axiom suggestions as well as their confidence score for each chosen axiom type in forms of tables. Additionally, it is possible to get some more details about the confidence score by clicking on the question mark symbol(?). This shows up a new dialog as shown in Fig. 8. The dialog gives some natural language based explanation about the F-score depending on the axiom type. Moreover, positive and negative examples (if exists) according to the axiom are shown, thus, giving some more detailed insights in how the axiom fits the data of the knowledge base.

2.3 Constraint Based Validation

2.3.1 Motivation

Integrity constraints provide a mechanism for ensuring that data conforms to guidelines specified by the defined schema. The demand for validating instance data as in relational databases or XML tools also holds for knowledge modeled in languages of the Semantic Web.


Explanation

Axiom: Functional: http://dbpedia.org/ontology/birthPlace

Score:98.58%

In **44365** triples where an individual *x* is connected by **http://dbpedia.org/ontology/birthPlace** to an individual *y*, there are **43739** cases where *y* is unique.

Positive examples:

http://dbpedia.org/resource/"Freeway"_Rick_Ross
http://dbpedia.org/resource/"Irish"_Teddy_Mann
http://dbpedia.org/resource/Ádám_Pintér
http://dbpedia.org/resource/Ádám_Présinger

Negative examples:

http://dbpedia.org/resource/Álvaro_Galindo
http://dbpedia.org/resource/Đorđe_Vujadinović
http://dbpedia.org/resource/İzzet_Türkyılmaz
http://dbpedia.org/resource/Ľudovít_Štúr

FUNCTIONAL OBJECT PROPERTY AXIOMS

	Accuracy	Axiom
☐	98.58% ?	**Functional:** dbo:birthPlace

Fig. 8. Screenshot of confidence score explanation in enrichment view for SPARQL knowledge bases.

In some use cases and for some requirements, OWL users assume and intend OWL axioms to be interpreted as Integrity Constraints. However, the direct semantics of OWL[14] does not interpret OWL axioms in this way; thus, the consequences that one can draw from such ontologies differ from the ones that some users intuitively expect and require. In other words, some users want to use OWL as a validation or constraint language for RDF instance data, but that is not possible using OWL based tools that correctly implement the standard semantics of OWL.

To see the nature of the problem, consider an OWL ontology that describes terms and concepts regarding a book store. The ontology includes the classes `Book` and `Writer`, the object property `hasAuthor`, and the data property `hasISBN`. Suppose we want to impose the following ICs on the data:

1. Each book must have an ISBN
2. Only books can have ISBNs
3. Books must not have more than one author

[14] http://www.w3.org/TR/owl2-direct-semantics/

These constraints could be interpreted in the following way:

Whenever an instance bookX of `Book` is added to the ontology, a check should be performed to verify whether the ISBN of bookX has been specified; if not, the update should be rejected. Whenever a fact `<bookX, hasISBN, ISBNX>` is added to the ontology, a check should be performed to verify whether bookX is an instance of `Book`; if not, the update should be rejected. Whenever a fact `<bookX, hasAuthor, writerX>` is added to the ontology, a check should be performed to verify whether another writer writerY has been specified for bookX; if so, the update should be rejected. These constraints can be concisely and unambiguously represented as OWL axioms:

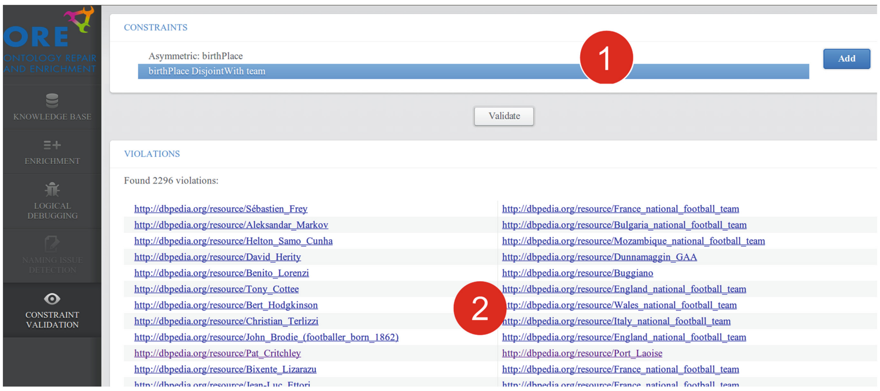
```
Class: Book
  SubClassOf: hasISBN some xsd:string         (axiom 1)
DataProperty: hasISBN
  Domain: Book                                (axiom 2)
ObjectProperty: hasAuthor
  Characteristics: Functional                 (axiom 3)
```

However, these axioms will not be interpreted as checks by tools which implement the standard OWL semantics. In fact, according to the standard OWL semantics, we have that:

1. Having a book without an ISBN in the ontology does not raise an error, but leads to the inference that the book in question has an unknown ISBN. (by axiom 1)
2. Having a fact `<bookA, hasISBN, ISBN1>` in the ontology without bookA being an instance of `Book` does not raise an error, but leads to the inference that bookA is an instance of `Book`. (by axiom 2)
3. Having a fact `<bookA, hasAuthor, writerA>` having specified a previous writer writerB for bookA does not raise an error, but leads to the inference that writerA and writerB denote the same individual. (by axiom 3)

In some cases, users want these inferences; but in others, users want integrity constraint violations to be detected, reported, repaired, etc.

One approach for using OWL as an expressive schema language, but giving it an alternative semantics such that OWL axioms can be used as ICs, was proposed in [20]. The idea behind it is to interpret OWL axioms with Closed World Assumption (CWA) and a weak form of Unique Name Assumption (UNA). Assuming a CWA interpretation basically means that an assertion is false if it is not explicitly known it is true or false. Weak UNA means that if two individuals are not inferred to be the same, then they will be assumed to be distinct. Based on these assumptions, translating an OWL axiom into one or more SPARQL queries is suggested to validate the given constraint. This approach is integrated in ORE, thus, it is possible to define and validate ICs by reusing OWL as a language.



CONSTRAINTS

Asymmetric: birthPlace
birthPlace DisjointWith team

VIOLATIONS

Found 2296 violations:

http://dbpedia.org/resource/Sébastien_Frey	http://dbpedia.org/resource/France_national_football_team
http://dbpedia.org/resource/Aleksandar_Markov	http://dbpedia.org/resource/Bulgaria_national_football_team
http://dbpedia.org/resource/Helton_Samo_Cunha	http://dbpedia.org/resource/Mozambique_national_football_team
http://dbpedia.org/resource/David_Herity	http://dbpedia.org/resource/Dunnamaggin_GAA
http://dbpedia.org/resource/Benito_Lorenzi	http://dbpedia.org/resource/Buggiano
http://dbpedia.org/resource/Tony_Cottee	http://dbpedia.org/resource/England_national_football_team
http://dbpedia.org/resource/Bert_Hodgkinson	http://dbpedia.org/resource/Wales_national_football_team
http://dbpedia.org/resource/Christian_Terlizzi	http://dbpedia.org/resource/Italy_national_football_team
http://dbpedia.org/resource/John_Brodie_(footballer_born_1862)	http://dbpedia.org/resource/England_national_football_team
http://dbpedia.org/resource/Pat_Critchley	http://dbpedia.org/resource/Port_Laoise
http://dbpedia.org/resource/Bixente_Lizarazu	http://dbpedia.org/resource/France_national_football_team
http://dbpedia.org/resource/Jean-Luc_Ettori	http://dbpedia.org/resource/France_national_football_team

Fig. 9. Screenshot of constraint validation view.

2.3.2 Support in ORE

Basically, the constraint validation view (see Fig. 9) consists of two parts. In the upper table (1) the user can define a list of constraints by adding OWL axioms, here for instance that the object properties `birthPlace` and `team` are disjoint, i.e. there are no pairs of instances that are related by both properties. The bottom part (2) is used to visualize violations of the given constraints. In the example on Fig. 9, it was found that `Pat Critchley` was born in `Port Laoise`, but also was a team member of it, which is obviously a contradiction to the disjointness statement.

3 Ontology Repair with PatOMat

The *PatOMat* is a pattern-based ontology transformation framework specifically designed for OWL ontologies [23]. By applying transformation it enables a designer to modify the structure of an ontology or its fragments to make it more suitable for a target application. While it can adapt any ontology aspect (logical, structural, naming or annotation aspect), within the context of LOD2 project the PatOMat focuses on ontology naming aspect.

During the decades of knowledge engineering research, there has been recurrent dispute on how the natural language structure influences the structure of formal knowledge bases and vice versa. A large part of the community seems to recognise that the content expressed in formal representation languages, such as the semantic web ones, should be accessible not only to logical reasoning machines but also to humans and NLP procedures, and thus resemble the natural language as much as possible [17].

Often, an ontology naming practice can be captured as a *naming pattern*. For instance, it is quite common in ontologies that a subclass has the same head noun as its parent class (Non-Matching Child Pattern).[15] By an earlier study [22] it was estimated that in ontologies for technical domains this simple pattern is verified in 50–80 % of class-subclass pairs such that the subclass name is a multi-token one. This number further increases if one considers thesaurus correspondence (synonymy and hypernymy) rather than literal string equality. In fact, the set-theoretic nature of taxonomic path entails that the correspondence of head nouns along this path should be close to 100 % in principle; the only completely innocent deviations from it should be those caused by incomplete thesauri. In other words, any violation of head noun correspondence may potentially indicate a (smaller or greater) problem in the ontology. Prototypical situations are:

- Inadequate use of class-subclass relationship, typically in the place of whole-part or class-instance relationship, i.e., a conceptualisation error frequently occurring in novice ontologies.
- Name shorthanding, typically manifested by use of adjective, such as "State-Owned" (subclass of "Company").

While the former requires complex refactoring of the ontology fragment, the latter can be healed by propagation of the parent name down to the child name.

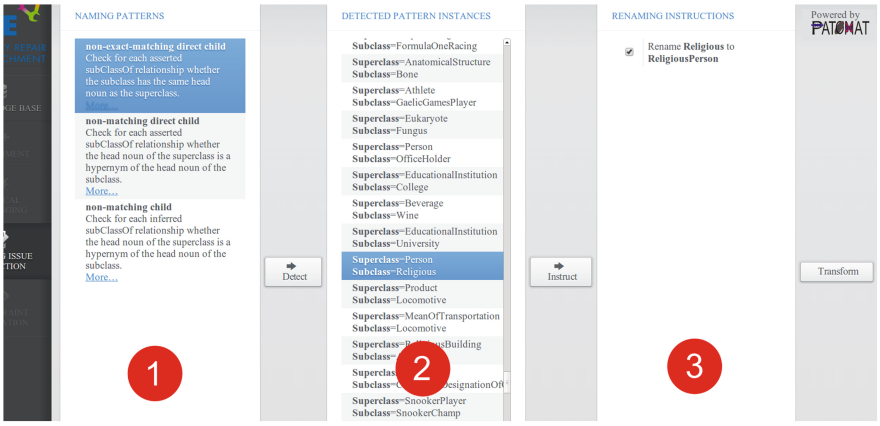
While in the biomedical field there have already been efforts in naming analysis, e.g., in [6,19], naming in the broad field of linked data vocabularies (where domain- specific heuristics cannot be applied) has rarely been addressed.

A pattern in the PatOMat framework, called transformation, consists of three parts: two *ontology patterns* (source OP and target OP) and the description of the transformation between them, called *pattern transformation* (PT). Naming pattern, such as non-matching child pattern, can be captured by specifying violation of a naming pattern to be detected (i.e. source OP) and its refactored variant (e.g. non-matching child pattern as target OP). Transformation patterns can be designed directly as XML files or by using graphical editor. For general usage the framework can be applied directly from the code by importing the PatOMat Java library[16] or by using *Graphical User Interface for Pattern-based Ontology Transformation* [21].

Naming issue detection and repair is supported by integrating the PatOMat framework into the ORE. The whole process is basically done in three subsequent steps, all of them visualized in a single view shown in Fig. 10. Here the user can select a naming pattern in the leftmost list (①). PatOMat then detects instances of the selected pattern in the currently loaded ontology, e.g. $[?OP1P = Contribution; ?OP1A = Poster]$(②). For the selected pattern instances the user will be provided a list of renaming instructions (see ③), for example to rename the class *Poster* to *PosterContribution*, which can then be used to transform the ontology and solve the detected naming issues.

[15] The head noun is typically the last token, but not always, in particular due to possible prepositional constructions, as, e.g., in "HeadOfDepartment".

[16] http://owl.vse.cz:8080/



non-exact-matching direct child
Check for each asserted subClassOf relationship whether the subclass has the same head noun as the superclass.
More...

non-matching direct child
Check for each asserted subClassOf relationship whether the head noun of the superclass is a hypernym of the head noun of the subclass.
More...

non-matching child
Check for each inferred subClassOf relationship whether the head noun of the superclass is a hypernym of the head noun of the subclass.
More...

Subclass=FormulaOneRacing
Superclass=AnatomicalStructure
Subclass=Bone
Superclass=Athlete
Subclass=GaelicGamesPlayer
Superclass=Eukaryote
Subclass=Fungus
Superclass=Person
Subclass=OfficeHolder
Superclass=EducationalInstitution
Subclass=College
Superclass=Beverage
Subclass=Wine
Superclass=EducationalInstitution
Subclass=University
Superclass=Person
Subclass=Religious
Superclass=Product
Subclass=Locomotive
Superclass=MeanOfTransportation
Subclass=Locomotive
Superclass=R[illegible]usBuilding
Subclass=[illegible]
Superclas[illegible]
Subclass=C[illegible]DesignationOf
Superclass=SnookerPlayer
Subclass=SnookerChamp

☑ Rename **Religious** to **ReligiousPerson**

Fig. 10. Screenshot of naming pattern detection and repair view in the ORE.

4 Linked Data Quality Assessment with RDFUnit

RDFUnit [10–12][17] is a framework for test-driven Linked Data quality assessment, which is inspired by test-driven software development. A key principle of test-driven software development is to start the development with the implementation of automated test-methods before the actual functionality is implemented. Compared to software source code testing, where test cases have to be implemented largely manually or with limited programmatic support, the situation for Linked Data quality testing is slightly more advantageous. On the Data Web we have a unified data model – RDF – which is the basis for both, data *and* ontologies. RDFUnit exploits the RDF data model by devising a pattern-based approach for the data quality tests of knowledge bases. Ontologies, vocabularies and knowledge bases can be accompanied by a number of test cases, which help to ensure a basic level of quality. This is achieved by employing SPARQL query templates, which are instantiated into concrete quality test SPARQL queries. We provide a comprehensive library of quality test patterns, which can be instantiated for rapid development of more test cases. Once test cases are defined for a certain vocabulary, they can be applied to all datasets reusing elements of this vocabulary. Test cases can be re-executed whenever the data is altered. Due to the modularity of the approach, where test cases are bound to certain vocabulary elements, test cases for newly emerging datasets, which reuse existing vocabularies can be easily derived.

RDFUnit is capable of performing quality assessments with only a minimal amount of manual user intervention and is easily applicable to large datasets. Other tools like the TopBraid Composer[18] use the *SPARQL Inferencing Notation*

[17] http://rdfunit.aksw.org
[18] www.topbraidcomposer.com

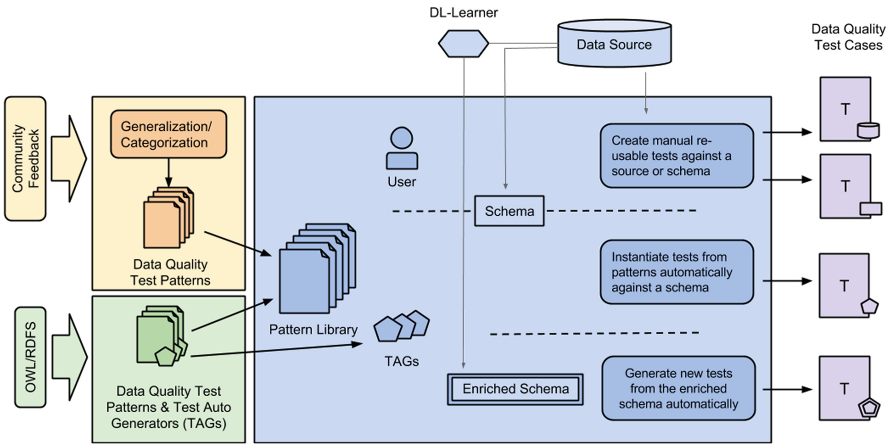

DL-Learner
Data Source
Data Quality Test Cases
Community Feedback
Generalization/ Categorization
Data Quality Test Patterns
User
Create manual re-usable tests against a source or schema
Schema
Instantiate tests from patterns automatically against a schema
Pattern Library
TAGs
OWL/RDFS
Data Quality Test Patterns & Test Auto Generators (TAGs)
Enriched Schema
Generate new tests from the enriched schema automatically
T

Fig. 11. Flowchart showing the test-driven data quality methodology. The left part displays the input sources of our pattern library. In the middle part the different ways of pattern instantiation are shown which lead to the data quality test cases on the right.

(SPIN)[19] to define SPARQL queries for quality assessment. However, RDFUnit utilizes an own SPARQL template notation, which better suits our methodology. An overview of the methodology is depicted in Fig. 11.

5 Analysis of Link Validity

5.1 Web Linkage Validator

With the integration of data into the LOD cloud, it is essential that links between datasets are discoverable as well as efficiently and correctly assessed. The Web Linkage Validator is a web-based tool that allows for knowledge base owners to improve their data with respect to linkage and to assess their linked data for integration with the LOD cloud.

The goal is to provide a tool to the LOD2 stack to aid in assessing links between LOD datasets. It analyses the links between entities that a dataset has as well as links to entities from other datasets. It will help knowledge base users in improving the quality of links of their datasets.

The Web Linkage Validator's assessment is based on the concept of a *data graph summary* [3,4]. A data graph summary is a concise representation of the RDF data graph and is composed of the structural elements, i.e., class and property. The information it contains, such as RDF class and predicate, usage frequency, provenance and linkage, are the basis for suggesting to knowledge base owners ways in which they may create or improve the links within their datasets and with other external datasets.

[19] http://spinrdf.org/

5.2 Data Graph Summary Model

In general, an RDF graph consists of datasets which in turn contain a number of entities. These entities are organised into classes. Links can exist at any of these levels; either between datasets, between class of entities or between the entities themselves. The data graph summary is a meta-graph that highlights the structure of a data graph (e.g. RDF).

For the graph summary process, we need to represent the data graph using three conceptual layers: the *dataset* layer, the *node collection* layer and the *entity* layer. The entity layer represents the original data graph. The node collection layer captures the schema and structure of the data graph in a concise way by grouping similar entities into a parent node that we call a *node collection.* This grouping is required as it allows for the graph summary to correctly determine collection specific information about those entities. The dataset layer captures the link structure across datasets as well as the provenance of the information on the entity and node collection layers. The Fig. 12 gives an example of the three layer representation of a data graph. Note that the $\star$ symbol represents terminating or leaf entities, e.g., RDF literal values. The node collection layer represents a summary computed by grouping together entities having the same classes. The node collection layer is composed of node collections and linksets, i.e., a set of links having the same labels between two node collections. For example, in the figure the links "author" between articles and people on the entity layer are mapped to two linksets "author" on the node collection layer.

5.3 Link Analysis

A data graph summary provides a unique view on the linkage information of a dataset. Using this meta-graph, it is possible to analyse the links of a dataset from the "point of view" of said dataset: links from/to other datasets, internal links between classes, etc. The Web Linkage Validator shown in Fig. 13a presents various "point of views" for the purpose of aiding the owners of datasets in the assessment of their linked data.

Besides giving a structural breakdown of the dataset, the graph summary is a utility for validating the internal and external links of a particular graph of data. In terms of external links, it shows what a dataset is "saying" about other datasets and vice-versa. This is important as it gives knowledge base owners the ability to validate what the links represent.

5.4 Provenance

The provenance represents the *origin* of entities and links. The provenance includes a classification of the relationships as *intra-dataset* or *inter-dataset* respectively based on entity linkage inside singular datasets or across multiple datasets. For example, a link between two entities can be classified as internal to a dataset because it was published within it, but can also be further classified as inter-dataset because the relationship contains an entity **outside** of the

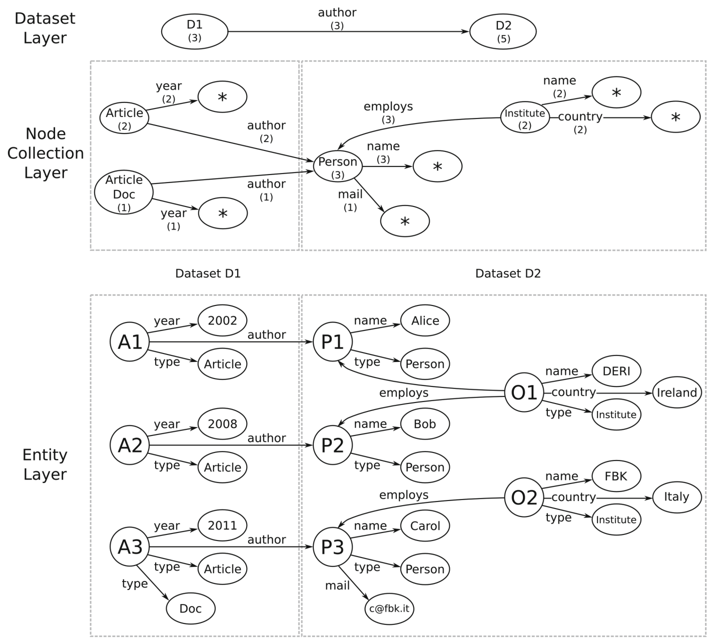


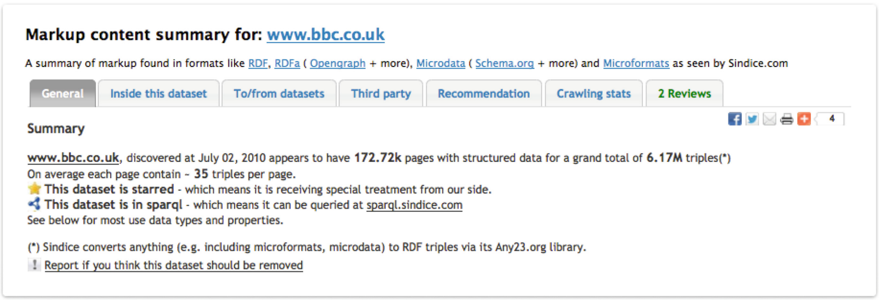
Fig. 12. A three layer representation of our Web Data model. On the node collection layer, nodes labelled with a star ⋆ represent blank collections.

publishing context. The Fig. 13b presents a view of the Web Linkage Validator showing the links internal to a dataset.

Direction. Inter-dataset triples are classified as *incoming* or *outgoing* depending on the direction of the link, relative to its origin (the subject entity) and to its destination (the object entity), based on its perspective (the context or publishing domain of the triple). For example, a triple published on the knowledge base (or domain) "xyz.com" that has a subject entity from "123.com" and object entity from "xyz.com" would be classified as incoming.

Authority. Similar to provenance and direction, the authority is based on the datasets and entities linked in a relationship. A link is classified as *authoritative* if at least one entity of the link originates from the publishing domain. For example, if a triple was published on "xyz.com" and the subject was from "xyz.com" and the object was from "123.com", then this link would be considered authoritative because "xyz.com" is asserting it. However, if the domain in which this triple was published was changed to "123.com", then it would become a non-authoritative link.

Third-Party Links. In regards to validating datasets, the authority classification helps knowledge base owners to distinguish another important aspect: third-party links. These represent non-authoritative links where both the subject and object of the link are defined in a dataset other than the publishing



Markup content summary for: www.bbc.co.uk

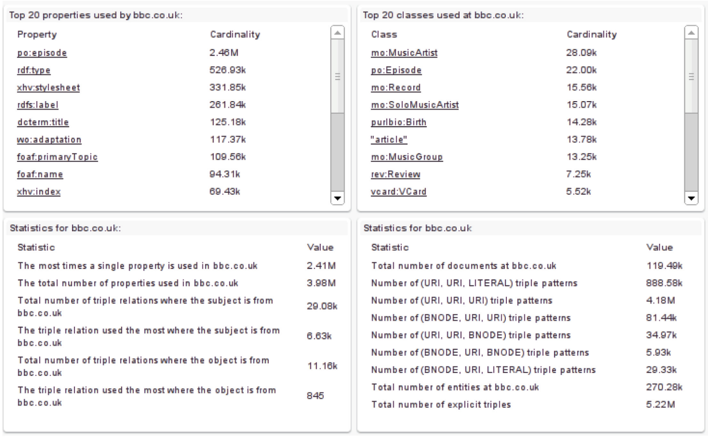
A summary of markup found in formats like RDF, RDFa (Opengraph + more), Microdata (Schema.org + more) and Microformats as seen by Sindice.com

Summary

www.bbc.co.uk, discovered at July 02, 2010 appears to have **172.72k** pages with structured data for a grand total of **6.17M** triples(*)
On average each page contain - **35** triples per page.
This dataset is starred - which means it is receiving special treatment from our side.
This dataset is in sparql - which means it can be queried at sparql.sindice.com
See below for most use data types and properties.

(*) Sindice converts anything (e.g. including microformats, microdata) to RDF triples via its Any23.org library.
Report if you think this dataset should be removed

(a) General view



Top 20 properties used by bbc.co.uk:

Property	Cardinality
po:episode	2.46M
rdf:type	526.93k
xhv:stylesheet	331.85k
rdfs:label	261.84k
dcterm:title	125.18k
wo:adaptation	117.37k
foaf:primaryTopic	109.56k
foaf:name	94.31k
xhv:index	69.43k

Top 20 classes used at bbc.co.uk:

Class	Cardinality
mo:MusicArtist	28.09k
po:Episode	22.00k
mo:Record	15.56k
mo:SoloMusicArtist	15.07k
purlbio:Birth	14.28k
"article"	13.78k
mo:MusicGroup	13.25k
rev:Review	7.25k
vcard:VCard	5.52k

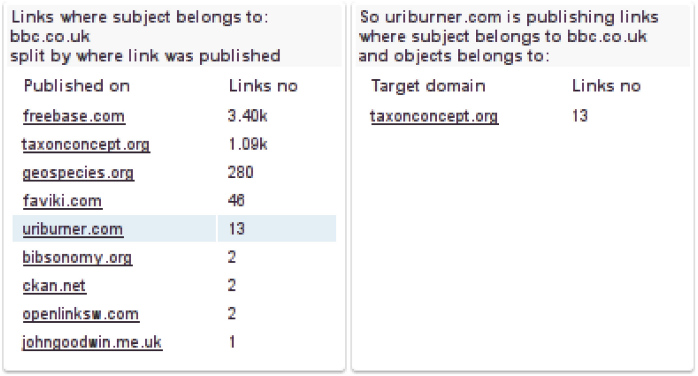
Statistics for bbc.co.uk:

Statistic	Value
The most times a single property is used in bbc.co.uk	2.41M
The total number of properties used in bbc.co.uk	3.98M
Total number of triple relations where the subject is from bbc.co.uk	29.08k
The triple relation used the most where the subject is from bbc.co.uk	6.63k
Total number of triple relations where the object is from bbc.co.uk	11.16k
The triple relation used the most where the object is from bbc.co.uk	845

Statistics for bbc.co.uk

Statistic	Value
Total number of documents at bbc.co.uk	119.49k
Number of (URI, URI, LITERAL) triple patterns	888.58k
Number of (URI, URI, URI) triple patterns	4.18M
Number of (BNODE, URI, URI) triple patterns	81.44k
Number of (URI, URI, BNODE) triple patterns	34.97k
Number of (BNODE, URI, BNODE) triple patterns	5.93k
Number of (BNODE, URI, LITERAL) triple patterns	29.33k
Total number of entities at bbc.co.uk	270.28k
Total number of explicit triples	5.22M

(b) Inside this dataset



Links where subject belongs to:
bbc.co.uk
split by where link was published

Published on	Links no
freebase.com	3.40k
taxonconcept.org	1.09k
geospecies.org	280
faviki.com	46
uriburner.com	13
bibsonomy.org	2
ckan.net	2
openlinksw.com	2
johngoodwin.me.uk	1

So uriburner.com is publishing links
where subject belongs to bbc.co.uk
and objects belongs to:

Target domain	Links no
taxonconcept.org	13

(c) Third-party links

Fig. 13. Views on a dataset provided by the Web Linkage Validator application.

one. Also, they are useful to discover if they consist of links that are incorrect or specify relationships that the owner does not explicitly agree with. In some cases, these links can be connotative to the idea of e-mail spam. Figure 13c presents the view of the Web Linkage Validator that provides information on the links classified as non-authoritative.

5.5 How to Improve Your Dataset with the Web Linkage Validator

In this section we show how the results of the Web Linkage Validator can be used as suggestions for improving one's dataset. Being able to see the classes and the properties of his dataset, the dataset owner is able to have a deep understanding of his dataset. He can determine if the dataset graph looks as he planned. For example, let's assume the dataset contains the "foaf:Person" class which has, among others, the "foaf:name" and "foaf:homepage" properties. From the number of the occurrences of these properties, the dataset owner can decide if his dataset is as he intended too: if he knows that most of the people in the dataset should have a homepage, then this should be reflected in similar numbers for the occurences of the "foaf:name" and "foaf:homepage" properties.

Also, the dataset owner can identify possible mistakes like typos in the classes/properties names. For example, it is well known that "foaf:name" is a property of the FOAF vocabulary but "foaf:naem" is not.

Moreover, having access to the number of links to and from other datasets, the dataset owner can determine whether his dataset really is part of the LOD. If the number of links to/from other datasets is quite small or even missing completely, the Web Linkage Validator supports the dataset owner in improving the dataset by suggesting similar datasets to which the dataset owner can link. Based on the top most similar dataset, the dataset owner identify concepts in the recommended dataset similar to the ones he uses and link them.

Once the changes have been done and the dataset has been improved, the dataset owner changes his website or his dataset dump. The infrastructure on which the Web Linkage Validator is based will recompute the data graph summary for the resubmitted dataset and next time the user will see his improvements.

6 Benchmarking Semantic Named Entity Recognition Systems

Named entity recognition (NER) became one of the most exploited means for information extraction and content enrichment. The NER systems detect text fragments identifying entities and provide classification of the entities into a set of pre-defined categories. This is usually a fixed set of raw classes such as the CoNLL set (PERSON, ORGANIZATION, LOCATION, MISCELLANEOUS), or classes from an ontology, such as the DBpedia Ontology. However, it is a recent trend that the NER systems such as DBpedia Spotlight to go beyond this type classification and also perform unique identification of the entities using

URIs from a knowledge bases such as DBpedia or Wikipedia. During LOD2, we have created a collection of tools adhering to this new class of *Wikification*, *Semantic NER* or *Entity Linking* systems and contributed it the Wikipedia page about Knowledge Extraction[20].

While these Semantic NER systems are gaining popularity, there is yet no oversight on their performance in general, and their performance in specific domains. To fill this gap, we have developed a framework for benchmarking NER systems [5][21]. It is developed as a stand-alone project on top of the GATE text engineering framework[22]. It is primarily developed for off-line evaluation of NER systems. Since different NER systems might perform better in one and worse in another domain, we have also developed two annotated datasets with entities, the News and the Tweets dataset. The Tweets datasets, consists of very large number of short texts (tweets), while the News dataset consists of standard-length news articles.

A prerequisite for benchmarking different NER tools is achieving interoperability at the *technical*, *syntactical* and *conceptual* level. Regarding the technical interoperability, most of the NER tools provide a REST API over the HTTP protocol. At the syntactical and conceptual level we opted for the NIF format, which directly addresses the syntactical and the conceptual aspects. The syntactical interoperability is addressed using the RDF and OWL as standards for common data model, while the conceptual interoperability is achieved by identifying the entities and the classes using global unique identifiers. For identification of the entities we opted for re-using URIs from DBpedia. Since different NER tools classify the entities with classes from different classification systems (classification ontologies), we perform alignment of those ontologies to the DBpedia Ontology[23].

In the future, we hope to exploit the availability of interoperable NIF corpora as described in [10].

7 Conclusion

In this chapter we have presented tools for conversion and extraction of data into RDF that were developed in the context of the LOD2 project. Specifically, the DBpedia Extraction Framework supports the extraction of knowledge from Wikis such as Wikipedia, the RDFa, Microdata and Microformats Extraction Framework crawls and collects data from the Web and Rozeta enables users to create and refine terminological data such as dictionaries and thesauri from natural language text. Once this data has been extracted and lifted to RDF, tools such as ORE and RDFUnit can analyse data quality and repair errors via a GUI. The presented tools are open source and make part of the Linked Data

[20] A frequently updated list can be found here http://en.wikipedia.org/wiki/Knowledge_extraction#Tools.

[21] http://ner.vse.cz/datasets/evaluation/

[22] http://gate.ac.uk/

[23] http://wiki.dbpedia.org/Ontology

stack (see Chap. 6). The tools have been extensively evaluated, for the details the reader is referred to the respective sections, cited articles and tools' webpages. These tools have been applied within LOD2 project, e.g. in a media publishing, enterprise and public procurement use cases, for the details see Chaps. 7, 8 and 10 of the present book, respectively.

References

1. Baader, F., Hollunder, B.: Embedding defaults into terminological knowledge representation formalisms. In: Nebel, B., Rich, C., Swartout, W.R., (eds.) KR, pp. 306–317. Morgan Kaufmann (1992)
2. Bizer, C., Eckert, K., Meusel, R., Mühleisen, H., Schuhmacher, M., Völker, J.: Deployment of RDFA, microdata, and microformats on the web - a quantitative analysis. In: Proceedings of the In-Use Track of the 12th International Semantic Web Conference (2013)
3. Campinas, S., Perry, T.E., Ceccarelli, D., Delbru, R., Tummarello, G.: Introducing RDF graph summary with application to assisted SPARQL formulation. In: 23rd International Workshop on Database and Expert Systems Applications, DEXA 2012, pp. 261–266, Sept 2012
4. Campinas, S., Delbru, R., Tummarello, G.: Efficiency and precision trade-offs in graph summary algorithms. In: Proceedings of the 17th International Database Engineering and Applications Symposium, IDEAS '13, pp. 38–47. ACM, New York (2013)
5. Dojchinovski, M., Kliegr, T.: Datasets and GATE evaluation framework for benchmarking wikipedia-based NER systems. In: Proceedings of 1st International Workshop on NLP and DBpedia, 21–25 October 2013, Sydney, Australia, volume 1064 of NLP & DBpedia 2013, Sydney, Australia, October 2013, CEUR Workshop Proceedings (2013)
6. Fernandez-Breis, J.T., Iannone, L., Palmisano, I., Rector, A.L., Stevens, R.: Enriching the gene ontology via the dissection of labels using the ontology pre-processor language. In: Cimiano, P., Pinto, H.S. (eds.) EKAW 2010. LNCS, vol. 6317, pp. 59–73. Springer, Heidelberg (2010)
7. Kalyanpur, A.: Debugging and repair of OWL ontologies. Ph.D. thesis, University of Maryland, College Park, College Park, MD, USA (2006) (Adviser-James Hendler)
8. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all justifications of OWL DL entailments. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 267–280. Springer, Heidelberg (2007)
9. Kontokostas, D., Bratsas, Ch., Auer, S., Hellmann, S., Antoniou, I., Metakides, G.: Internationalization of linked data: the case of the greek DBpedia edition. Web Semant. Sci. Serv. Agents World Wide Web **15**, 51–61 (2012)

10. Kontokostas, D., Brümmer, M., Hellmann, S., Lehmann, J., Ioannidis, L.: NLP data cleansing based on linguistic ontology constraints. In: Proceedings of the Extended Semantic Web Conference 2014 (2014)
11. Kontokostas, D., Westphal, P., Auer, S., Hellmann, S., Lehmann, J., Cornelissen, R.: Databugger: a test-driven framework for debugging the web of data. In: Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web Companion, WWW Companion '14, pp. 115–118, Republic and Canton of Geneva, Switzerland, 2014, International World Wide Web Conferences Steering Committee
12. Kontokostas, D., Westphal, P., Auer, S., Hellmann, S., Lehmann, J., Cornelissen, R., Zaveri, A.: Test-driven evaluation of linked data quality. In: Proceedings of the 23rd International Conference on World Wide Web, WWW '14, pp. 747–758, Republic and Canton of Geneva, Switzerland, 2014, International World Wide Web Conferences Steering Committee
13. Lehmann, J., Bizer, Ch., Kobilarov, G., Auer, S., Becker, Ch., Cyganiak, R., Hellmann, S.: DBpedia - a crystallization point for the web of data. J. Web Semant. **7**(3), 154–165 (2009)
14. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. Semant. Web J. (2014)
15. Mika, P., Potter, T.: Metadata statistics for a large web corpus. In: LDOW: Linked Data on the Web. CEUR Workshop Proceedings, vol. 937 (2012)
16. Mühleisen, H., Bizer, C., Web data commons - extracting structured data from two large web corpora. In: LDOW: Linked Data on the Web. CEUR Workshop Proceedings, vol. 937 (2012)
17. Nirenburg, S., Wilks, Y.: What's in a symbol: ontology, representation and language. J. Exp. Theor. Artif. Intell. **13**(1), 9–23 (2001)
18. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence, pp. 355–360. Morgan Kaufmann Publishers, San Francisco (2003)
19. Schober, D., Smith, B., Lewis, S.E., Kusnierczyk, W., Lomax, J., Mungall, C., Taylor, C.F., Rocca-Serra, P., Sansone, S.-A.: Survey-based naming conventions for use in OBO foundry ontology development. BMC Bioinform. **10**(1), 125 (2009)
20. Sirin, E., Tao, J.: Towards integrity constraints in OWL. In: Hoekstra, R., Patel-Schneider, P.F., (eds.) OWLED, volume 529 of CEUR Workshop Proceedings (2008). http://CEUR-WS.org
21. Šváb-Zamazal, O., Dudáš, M., Svátek, V.: User-friendly pattern-based transformation of OWL ontologies. In: ten Teije, A., Völker, J., Handschuh, S., Stuckenschmidt, H., d'Acquin, M., Nikolov, A., Aussenac-Gilles, N., Hernandez, N. (eds.) EKAW 2012. LNCS, vol. 7603, pp. 426–429. Springer, Heidelberg (2012)
22. Šváb-Zamazal, O., Svátek, V.: Analysing ontological structures through name pattern tracking. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008. LNCS (LNAI), vol. 5268, pp. 213–228. Springer, Heidelberg (2008)
23. Zamazal, O., Svátek, V.: Patomat - versatile framework for pattern-based ontology transformation. Comput. Inf. (2014) (Accepted)