Large-Scale Multilingual Knowledge Extraction, Publishing and Quality Assessment: The case of DBpedia

Der Fakultät für Mathematik und Informatik
der Universität Leipzig
eingereichte

DISSERTATION

zur Erlangung des akademischen Grades

Doktor-Ingenieur
(Dr.-Ing.)

im Fachgebiet
Informatik

vorgelegt

von **M.Sc. Dimitrios Kontokostas**

geboren am 06. Juni 1981 in Veria, Griechenland

Die Annahme der Dissertation wurde empfohlen von:

1. Prof. Dr. Klaus-Peter FÃ€hnrich, UniversitÃ€t Leipzig
2. Prof. Dr. Michel Dumontier, Maastricht University

Die Verleihung des akademischen Grades erfolgt mit Bestehen der Verteidigung am
02.05.2018 mit dem GesamtprÃ€dikat
magna cum laude.

Large-Scale Multilingual Knowledge Extraction, Publishing and Quality Assessment: The case of DBpedia

Der Fakultät für Mathematik und Informatik
der Universität Leipzig
eingereichte

DISSERTATION

zur Erlangung des akademischen Grades

Doktor-Ingenieur
(Dr.-Ing.)

im Fachgebiet
Informatik

vorgelegt

von **Dipl.-Inf. Dimitrios Kontokostas**

geboren am 06. Juni 1981 in Veria, Griechenland

Leipzig, den 22.3.2017

DIMITRIOS KONTOKOSTAS

LARGE-SCALE MULTILINGUAL KNOWLEDGE EXTRACTION, PUBLISHING AND QUALITY ASSESSMENT: THE CASE OF DBPEDIA

AUTHOR:
Dipl. Inf. Dimitrios Kontokostas

TITLE:
Large-Scale Multilingual Knowledge Extraction, Publishing and Quality Assessment: The case of DBpedia

INSTITUTION:
Institut für Informatik, Fakultät für Mathematik und Informatik, Universität Leipzig

BIBLIOGRAPHIC DATA:
2015, XX, 211p., 35 illus. in color., 34 tables

SUPERVISORS:
Prof. Dr. Klaus-Peter Fähnrich
Dr.-Ing. Sebastian Hellmann
Prof. Dr. Jens Lehmann

Dedicated to Aleka, my loving and supporting wife

THESIS SUMMARY

Title:
Large-Scale Multilingual Knowledge Extraction, Publishing and Quality Assessment: The case of DBpedia
Author:
Dimitrios Kontokostas
Bib. Data:
2015, XX, 211p.
35 illus. in color.
34 tables

Since its inception in 2001, Wikipedia has grown into one of the most widely used multilingual encyclopedias, covering more than 200 languages and being one of the finest examples of truly collaboratively created content. Besides of free text, Wikipedia articles consist of different types of (semi-) structured data such as infoboxes, tables, lists, and categorization data.

The DBpedia project builds a large-scale, multilingual knowledge base by extracting such structured data from Wikipedia editions in 130 languages. This knowledge base can be used to answer expressive queries that otherwise could not have been answered using only free text search. Being multilingual and covering a 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. This interdisciplinary and multilingual nature of the data enabled DBpedia to become a central hub for the Linked Open Data (LOD) cloud since 2007 and remain one of the central hubs for the web of data until now.

The latest release of DBpedia (v2016-04) consists of 9.5 billion facts. The DBpedia datasets have become the foundation of a plethora of academic and industrial projects. At the time of writing, around 18500 scientific articles that mention DBpedia in their text are published, out of which, 2700 published in 2016 (based on Google Scholar). Besides research, DBpedia is used for a wide range of industrial applications, verified by the number of company participation and presentations in the recent DBpedia meetings, as well as activity in the different DBpedia related discussion fora. This indicates the outreach of DBpedia to be broad.

However, the task of extracting facts from (semi-)structured or unstructured data is hard and involving many sub-tasks. This is amplified in crowd-sourced environments like Wikipedia, where there is no strict coordination for uniformity of the data. The variation is further increased when different Wikipedia language editions (for example Wikipedia in German or in Dutch) or other Wikimedia projects (for example Wikimedia Commons or Wikidata) are taken into account. These projects are managed from diverse communities and most of the times use different conventions that increase the extraction complexity.

Data quality comes at the intersection of knowledge extraction and usefulness, namely the results of an extraction process can only be evaluated with it's usefulness in a specific context. This makes quality mainly a context-specific metric and *fitness for use*. There is a huge list

of requirements one may have for extracted data, i.e. schema consistency, exhaustive coverage, correctness. For example, one may want the birth date of every person to be a date represented in UTC and in a specified format (i.e. *yyyy-mm-dd*) to avoid different representation formats or day and month conventions. Apart from the schema, there can be other types of (logical) constraints that people want to enforce on the data. An example can be that the birth date of a person should not be after her death place or that a person cannot have two birth dates. At the time of writing, except from some OWL features, there was no standardized way to define data schemas and constraints for RDF data validation purposes. People were usually using ontologies and RDF query languages to achieve such goals, even though they were not designed for this purpose. Finally, although identifying quality issues in small-in-size datasets is a hard task, scaling this problem for DBpedia – where billions of facts need to be verified – becomes even more challenging.

In this thesis there is a focus on both large-scale multilingual knowledge extraction and quality assessment since evaluating the good usability of the extracted knowledge is essential. This is is achieved by extending existing and adding new extractors in the *DBpedia information extraction framework*, improving internationalization and localization support, as well as by assessing the quality of DBpedia and enabling validation steps throughout the extraction process. The quality assessment tools and methods that were developed have been generalized and are applicable outside of DBpedia, to any RDF dataset.

LARGE-SCALE MULTILINGUAL KNOWLEDGE EXTRACTION & PUBLISHING The DBpedia project started long before the beginning of this thesis. The early versions of DBpedia put greater emphasis on the English Wikipedia as it is the most abundant language edition. During the extraction process, however, language-specific information was lost or ignored. Working on improving the internationalization and multilinguality in DBpedia resulted in an increase up to 85% in localized content and new specifications for publishing Unicode-enabled IRI identifiers.

Besides different Wikipedia-language editions there existed other Wikimedia projects that were proven to be a great source of information like Wikimedia Commons and Wikidata. Wikimedia Commons is the media backend of all Wikipedias. It is the central place where media files are uploaded under an open-access license and are easily referenced from articles in any Wikipedia language. Wikidata aims at becoming the structured data backend of Wikimedia, where people can reference facts and share them in articles in any Wikipedia language, as well as beyond Wikimedia projects. Both Wikimedia Commons and Wikidata were incorporated into the DBpedia data stack

and increased the resources defined by DBpedia by 40 million new identifiers.

With regard to data extraction there was additionally work on classifying DBpedia resources to types by leveraging the Wikipedia categories and *Natural Language Processing* (NLP) techniques. Although this thesis tried to identify and extract structured or semi-structured information, there is still a lot of room left for the extraction of unstructured information through NLP.

QUALITY ASSESSMENT In order to improve the quality of the DBpedia data, we focused on the creation of data quality assessment methodologies. A first attempt on assessing the quality of DBpedia was made with a crowd-sourced approach, where semantic web experts were asked to evaluate DBpedia facts. The evaluation was assisted by a custom developed tool called TripleCheckMate that guided the evaluators to identify and classify errors. The assessment results were very encouraging and helped in identifying violation clusters in DBpedia. However, it was an approach that could not easily scale to evaluate millions of facts.

The *Test-Driven Quality Assessment Methodology* (TDQAM) was another approach to bring software engineering methodologies like unit-testing in data and knowledge engineering. RDFUnit is a tool that was developed to implement this methodology. Taking advantage of the data model of RDF and ontologies, an automated data test-generation approach was conveyed to ease the validation task. Although this methodology was created with the purpose to assess DBpedia, the approach was general enough and could be applied to any RDF Dataset of any size. Moreover, it was shown that TDQAM is able to quickly surpass custom validation approaches developed by dataset or ontology maintainers.

Two additional domain specific adaptations of TDQAM were enabled through this thesis. In the first approach, the assessment was conducted directly in the mappings that generate the RDF trying to identify mappings that would always result in RDF with schema inconsistencies. The second approach integrated TDQAM and RDFUnit in software engineering workflows (i.e. Continuous Integration) and automatically performed a validation on every change in the code or the data transformation configurations. Whenever an inconsistency was detected the project fails, requesting an immediate repair from the person who introduced the failure.

A major impact of this thesis was the influence of SHACL, language for defining constraints on RDF graphs. SHACL is close to becoming a W3C recommendation and filling the existing data validation gap in the RDF technology stack. Finally, intermediate results of this thesis formed the basis of the *Data Quality* work package in the ALIGNED project.

In summary, this thesis forms a set of research and engineering results for improving the state of the art in *large-scale multilingual knowledge extraction, publishing and quality assessment*, with a focus on DBpedia. The results of this thesis are already contributed back to the scientific & industry community through an improved DBpedia open data stack and open source tools, services and specifications.

PUBLICATIONS

This thesis is based on the following publications, books and proceedings, in which I have been author, editor or contributor. *At the respective margin of each chapter and section*, I included the references to the appropriate publications.

Citations at the margin were the basis for the respective sections or chapters.

STANDARDS

- Editor of the upcoming W3C standard about *SHApes Constraint Lanugage* (SHACL), a major result of the work described here. Knublauch and Kontokostas, (2016)

PROCEEDINGS, (CO)-EDITED

- Proceedings of the 3nd Workshop on Linked Data Quality (LDQ) in conjunction with 13th Extended Semantic Web Conference (ESWC 2016) Heraklion, Crete, Greece, May 2016 Debattista, Umbrich, Fernandez, Rula, Zaveri, Knuth, and Kontokostas, (2016)
- Proceedings of the 2nd Workshop on Linked Data Quality (LDQ) in conjunction with 12th Extended Semantic Web Conference (ESWC 2015) Portoroz, Slovenia, June 2015 Rula, Zaveri, Knuth, and Kontokostas, (2015)
- Proceedings of the 1st Workshop on Linked Data Quality (LDQ) in conjunction with the 10th International Conference on Semantic Systems (SEMANTiCS 2014) Leipzig, Germany, September 2014 Knuth, Kontokostas, and Sack, (2014)
- Proceedings of the NLP and DBpedia Workshop in conjunction with the 12th International Semantic Web Conference (ISWC 2013) Sydney, Australia, October, 2013. Hellmann, Filipowska, Barriere, Mendes, and Kontokostas, (2013b)

JOURNAL PUBLICATIONS, PEER-REVIEWED

- Wikidata through the Eyes of DBpedia (under review). Ismayilov, Kontokostas, Auer, Lehmann, and Hellmann, (2016)
- DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. Lehmann, Isele, Jakob, Jentzsch, Kontokostas, Mendes, Hellmann, Morsey, Kleef, Auer, and Bizer, (2015)
- Multilingual Linked Data Patterns. Gayo, Kontokostas, and Auer, (2013)

- Internationalization of Linked Data: The case of the Greek DBpedia edition. Kontokostas, Bratsas, Auer, Hellmann, Antoniou, and Metakides, (2012)

CONFERENCE PUBLICATIONS, PEER-REVIEWED

- Semantically Enhanced Quality Assurance in the JURION Business Use Case Kontokostas, Mader, Dirschl, Eck, Leuthold, Lehmann, and Hellmann, (2016)
- DBpedia Commons: Structured Multimedia Metadata from the Wikimedia Commons Vaidya, Kontokostas, Knuth, Lehmann, and Hellmann, (2015)
- Assessing and Refining Mappings to RDF to Improve Dataset Quality Dimou, Kontokostas, Freudenberg, Verborgh, Lehmann, Mannens, Hellmann, and Walle, (2015)
- Unsupervised Learning of an Extensive and Usable Taxonomy for DBpedia Fossati, Kontokostas, and Lehmann, (2015)
- DataID: Towards Semantically Rich Metadata for Complex Datasets. Brümmer, Baron, Ermilov, Freudenberg, Kontokostas, and Hellmann, (2014)
- NLP data cleansing based on Linguistic Ontology constraints. Kontokostas, Brümmer, Hellmann, Lehmann, and Ioannidis, (2014b)
- Test-driven Evaluation of Linked Data Quality. Kontokostas, Westphal, Auer, Hellmann, Lehmann, Cornelissen, and Zaveri, (2014c)
- Towards Web Intelligence Through the Crowdsourcing of Semantics. Auer and Kontokostas, (2014)
- DBpedia Viewer - An Integrative Interface for DBpedia leveraging the DBpedia Service Eco System. Lukovnikov, Stadler, Kontokostas, Hellmann, and Lehmann, (2014)
- Crowdsourcing Linked Data quality assessment Acosta, Zaveri, Simperl, Kontokostas, Auer, and Lehmann, (2013)
- User-driven Quality Evaluation of DBpedia. Zaveri, Kontokostas, Sherif, Bühmann, Morsey, Auer, and Lehmann, (2013)

BOOK CHAPTERS

- Linked Open Data - Creating Knowledge Out of Interlinked Data. Hellmann, Bryl, Bḧmann, Dojchinovski, Kontokostas, Lehmann, Milošević, Petrovski, Svátek, Stanojević, and Zamazal, (2014)

DEMO & POSTER PUBLICATIONS, PEER-REVIEWED

- The DBpedia Data Stack – Towards a sustainable DBpedia Project to provide a public data infrastructure for Europe. Kontokostas and Hellmann, (2014)

- Databugger: A Test-driven Framework for Debugging the Web of Data. Kontokostas, Westphal, Auer, Hellmann, Lehmann, and Cornelissen, (2014a)
- TripleCheckMate: A Tool for Crowdsourcing the Quality Assessment of Linked Data. Kontokostas, Zaveri, Auer, and Lehmann, (2013)
- Towards Linked Data Internationalization - Realizing the Greek DBpedia. Kontokostas, Bratsas, Auer, Hellmann, Antoniou, and Metakides, (2011)
- DBpedia internationalization-a graphical tool for I18n infobox-to-ontology mappings. Bratsas, Ioannidis, Kontokostas, Auer, Bizer, Hellmann, and Antoniou, (2011a)

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest thanks to Dr. Sebastian Hellmann, Prof. Sören Auer and Prof. Klaus-Peter Fähnrich for giving me the opportunity to pursue my PhD at the university of Leipzig from my home in Greece and their continuous trust and support. Of course, this would not have been made possible without Enno Meijers, Gerard Kuys, Roland Cornelissen and the former *bibliotheek.nl* organization that funded the first two years of my PhD.

Furthermore, I would like to express my gratitude to Sebastian and Jens for being great mentors and available whenever I needed help and advice. I am even more thankful to Sebastian, for guiding me on how to be more effective, getting me involved in many interesting projects and always giving me new responsibilities.

I would like to thank all my colleagues who contributed directly or indirectly on this thesis. A big share is attributed to the AKSW group as a whole, highlighting Sebastian, Jens, Sören, Axel, Michael, Nadine, Amrapali, Claus, Seebi, Patrick and Lorenz. This list continues with our KILT subgroup, highlighting Sebastian, Milan, Kay, Julia, Markus and Martin. I would also like to thank my colleagues from the ALIGNED project for their feedback and ideas related to topics of this thesis.

With regard to the DBpedia community, there are numerous people who provided feedback in the discussion forums and the eight DBpedia meetings we organized. Amongst many others, I mention Magnus Knuth, Marco Fossati, Pablo Mendes, Joackim Daiber and Monika Solanki.

I would like to thank Agata Filipowska, Amrapali Zaveri, Anisa Rula, Caroline Barriere, Harald Sack, Jurgen Umbrich, Magnus Knuth, Pablo Mendes and Sebastian Hellmann, with whom we jointly organized the *NLP & DBpedia* (2013), *Linked Data Quality* (2014, 2015, 2016) workshops and the Semantic Web Journal *Special Issue on Quality Management of Semantic Web Assets (Data, Services and Systems)*. Furthermore, I would like to thank all the authors who submitted in those venues and the program committee members for their valuable help in reviewing these submissions.

I am thankful to all the discussions I had on DBpedia, W3C and quality assessment related mailing lists.

I would like to thank Patrick Westphal, for brainstorming with me the initial design of RDFUnit (called Databugger at that point in time), John McCrae for his support regarding the lemon vocabulary, Anastasia Dimou for her support with the assessment of RML mappings

and, Christian Dirschl (and the WKD dev-team) for the integration of RDFUnit with JUnit.

This thesis was partially supported by grant from the European Union's FP7 and H2020 programme provided for the projects ALIGNED (GA no. 644055), LIDER (GA no. 610782), LOD2 (GA no. 257943) and GeoKnow (GA no. 318159) and the former *bibliotheek.nl* organization.

I would like to thank Google and, in particular, the Google Summer of Code (GSoC) project. Besides a great experience, it got me in contact with great students and mentors; some of which, like Gaurav Vaidya, Denis Lukovnikov and Marco Fossati, became officially part of this thesis.

I am grateful to Nadine Jänicke, Cornelia Wimmer, Ulrike Weber, Julia Holze, Sandra Praetor, Anne Martin and many other colleagues for helping me fill out numerous German forms and translate documents.

And finally, I would like to thank my beloved wife Aleka, for her continuous support, patience and faith. This PhD would have not been possible without her. Of course, I should not forget Christina, our daughter, and apologize for our moments that I missed due to my traveling; Coincidently, Christina was an early result of this thesis.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

Part I

INTRODUCTION AND PRELIMINARIES

1 INTRODUCTION

Since its inception in 2001, Wikipedia has grown into one of the most widely used multilingual encyclopedia covering more than 200 languages and being one of the finest examples of truly collaboratively created content. At the same period, the semantic web (Berners-Lee, Hendler, and Lassila, 2001a) aims at the evolution of the World Wide Web towards representing the meaning of information in a way that is processable by machines. Recently, the Semantic Web vision was enriched by the concept of linked data (Bizer, Heath, and Berners-Lee, 2009), a movement which within short time led to a vast amount of linked data on the Web accessible in a simple yet standardised way.

DBpedia (Lehmann et al., 2009) sits at the crossroad of Wikipedia, semantic web and linked data. DBpedia is an effort to extract knowledge from Wikipedia, represents it as RDF, interlink it with other sources and publish the extracted knowledge according to the linked data principles. Wikipedia articles, besides of free text, contain different additional types of (semi) structured information such as infoboxes, tables, lists, and categorization data. The DBpedia project builds a large-scale, multilingual knowledge graph by extracting such information from Wikipedia editions in many languages. This is performed by the *DBpedia Information Extraction Framework* (DIEF), a flexible and pluggable framework that takes a Wikimedia project as an input source and generates a knowledge graph as output.

The DBpedia knowledge graph can be used to answer expressive queries that otherwise could not have been answered using only free-text search. For example, questions like *Which are the European countries with capitals populated by more than 5 million people* are now easy to formulate over Wikipedia. Being multilingual and covering a wide range of topics, the DBpedia knowledge graph is also useful within further application domains such as data integration, named entity recognition, topic detection, and document ranking. This interdisciplinary and multilingual nature of the data enabled DBpedia to become one of the most prominent linked data dataset examples as well as a central hub for the Linked Open Data (LOD) cloud since 2007, and, remain one of the central hubs for the web of data until now (Kobilarov et al., 2009).

The latest release of DBpedia (v2016-04[1]) consists of 9.5 billion facts (RDF triples) out of which 1.3 billion were extracted from the English edition of Wikipedia, 5.0 billion were extracted from other 129 language editions and 3.2 billion from Wikimedia Commons and Wiki-

1 http://wiki.dbpedia.org/dbpedia-version-2016-04

data. The DBpedia datasets have become the foundation of a plethora of academic and industrial projects. At the time of writing, DBpedia has very broad dissemination channels and around 1400 people are directly subscribed in DBpedia related lists. With regard to research, there are around 18500 scientific articles published that mention DBpedia in their text,[2] out of which, 2700 published in 2016[3]. Furthermore, as far as industry is concerned, DBpedia is used for a wide range of industrial applications, verified by the number of company participation and presentations in the recent DBpedia meetings as well as activity in the different DBpedia related discussion fora. This indicates the outreach of DBpedia to be broad.

However, the task of extracting knowledge from (semi-)structured or unstructured data is hard, involving many sub-tasks (Cunningham, 2005). Evaluating the results of knowledge extraction is also a difficult task and a common evaluation metric is precision, while recall (Olson and Delen, 2008). Precision in this context is defined as the fraction of extracted knowledge that is correct and recall as the fraction of available knowledge that is extracted. There is an inverse relationship between precision and recall, where it is possible to increase one at the cost of reducing the other.

Information in Wikipedia is curated by very diverse and highly dynamic communities (Roth, Taraborelli, and Gilbert, 2008) and there is no strict global coordination for uniformity of the data. This is amplified when different Wikipedia language editions (for example Wikipedia in German or in Dutch) or other Wikimedia projects (for example Wikimedia Commons or Wikidata) are taken into account. These projects are managed by separate communities and most of the times use different conventions for defining information that ranges from structural to language-specific conventions (for example locale settings for numbers and dates). All these factors further increase the complexity of improving the precision and recall in knowledge extraction from Wikipedia.

Data quality comes at the intersection of knowledge extraction and usefulness and provides a way to assess precision. However, the results of an extraction process can only be evaluated on the basis of its usefulness in a specific context, i.e. *fitness for use* Juran, 1974. This makes quality mostly a context specific metric. For example, the same data can be inappropriate for a medical application while sufficient for a simple web application. There is a huge list of requirements one may have for extracted data, i.e. schema consistency, exhaustive coverage, correctness, and many others. A very common use case is schema consistency, meaning to validate the data and assess whether all or parts of a dataset conform to a given schema. For instance, one

2 Accessed March 7th 2017 https://scholar.google.com/scholar?q=dbpedia

3 Accessed March 7th 2017 https://scholar.google.com/scholar?q=dbpedia&as_ylo=2016

may want the birth date of every person to be a date represented in UTC and in a specified format (i.e. *yyyy-mm-dd*) to avoid different representation formats or day and month conventions. Apart from the schema, there can be other types of (logical) constraints that people want to enforce on the data. An example can be that the birth date of a person should not be after her death place or that a person cannot have two birth dates. At the time of writing, except from some OWL features, there was no standardized way to define data schemas and constraints for RDF data validation purposes. [4] People were usually using ontologies and RDF query languages such as SPARQL for validation, even though ontologies and query languages were not designed for validation purposes. Finally, assessing the quality of datasets like DBpedia where billions of facts need to be evaluated, becomes an even more challenging problem, as tools that perform the assessment need to be able to scale.

This thesis forms a set of research and engineering results for increasing both precision and recall in *large-scale multilingual knowledge extraction*, with a focus on DBpedia. Increasing recall is achieved by extending existing and adding new extractors in the *DBpedia information extraction framework*, as well as improving internationalization and localization support. Increasing precision on the other hand ,is achieved by assessing the quality of DBpedia and enabling validation steps throughout the extraction process. Finally, the quality assessment methods that were developed have been generalized and are applicable outside of DBpedia, to any RDF dataset.

OVERVIEW

The structure of this thesis is the following

PART I INTRODUCTION This part of the thesis (Part i) is divided in two chapters: Introduction (Chapter 1) and Preliminaries (Chapter 2). The current section, Introduction, presents an overview of this thesis. In the following section, Preliminaries, definitions and background information for the main concepts of this thesis (i.e. *RDF* and *data quality*) are defined.

PART II - LARGE-SCALE MULTILINGUAL KNOWLEDGE EXTRACTION & PUBLISHING: THE CASE OF DBPEDIA Overall this part overviews DBpedia and describes major extensions performed as part of this thesis. Chapter 3, *DBpedia Overview*, provides an overall description of DBpedia. More emphasis is given on the *DBpedia information Extraction Framework* (DIEF) and its modular architecture and the crowdsourced enabled DBpedia ontology and Wikipedia Infoboxes to

4 The Shapes Constraint Language (SHACL) is currently in the standardization progress

DBpedia ontology mappings. A list of related *DBpedia extensions* with regard to data publishing, entity classification and dataset metadata definition is also described.

Early versions of DBpedia were focusing mainly on the English Wikipedia as the most abundant language edition. However, during the extraction process, language specific information was lost or ignored. Chapter 4, *Internationalization of DBpedia*, describes the process of extending DIEF to better support content from non-English Wikipedia language editions and the way a data increase of up to 85% per language was reached. In addition, it defines how Linked Data can be served with de-referencable IRIs, which was not possible before.

In addition to Wikipedia language editions, two new sources were identified and incorporated in the DBpedia data stack: Wikimedia Commons (cf. Chapter 5) and Wikidata (cf. Chapter 6). Wikimedia Commons forms the media backend of all Wikimedia projects and stores open access files such as images, videos, books, etc. DBpedia Commons is a semantic mirror of Wikimedia Commons using more than 1.4 billion RDF triples on file metadata, provenance, descriptions, and license information. Similar to Wikimedia Commons, Wikidata is a recent Wikimedia project with the aim to provide the structured data backend of all Wikimedia projects. The DBpedia-to-Wikidata dataset transforms the Wikidata data to the DBpedia ontology by employing a new mapping language. Even though Wikidata recently provides its data as RDF we argue that our approach is beneficial for both projects. Finally, Chapter 7 gives an overview of work related to knowledge graphs and knowledge extraction from Wikipedia.

PART III - LINKED DATA QUALITY ASSESSMENT In this part we define two methodologies for assessing the quality of RDF and Linked Data. Chapter 8 describes the *Test-Driven Quality Assessment Methodology* (TDQAM). TDQAM comprises of a core methodology, a workflow and an accompanied RDF vocabulary. In the core methodology concepts like *Data Quality Test Case*, *Data Quality Test Pattern*, *Test Case Pattern Binding*, *Test case Auto Generator* and *Test Coverage* are defined. The TDQAM workflow defines the different means of *Data Quality Test Case* and *Data Quality Test Pattern* elicitation for assessing an RDF dataset and an extensive reusable pattern library. Additionally, an RDF vocabulary for describing all the TDQAM concepts and workflows in RDF is included.

The *Test-Driven Quality Assessment Methodology* is thoroughly evaluated in Chapter 9. RDFUnit is a tool that was build to implement TDQAM. Using RDFUnit and domain experts we 1) automatically generated 32K reusable test cases for 297 RDF vocabularies, 2) evaluated the quality of five LOD datasets by reusing these test cases and

revealed a substantial amount of data quality issues in an effective and efficient way and, 3) quickly improved existing domain-specific validation in the NLP domain.

Chapter 10 (Crowdsourcing Quality in RDF) presents an additional methodology for assessing the quality of RDF with domain experts and a micro-task strategy. After defining the methodology, we describe TripleCheckMate, a tool that implements the methodology. Using TripleCheckMate, we created a DBpedia evaluation campaign where 58 domain experts evaluated a total of 792 DBpedia resources and identified 2928 incorrect triples. We provide and discuss the results of this work. Finally, Chapter 11 describes work related to RDF quality assessment.

PART IV - TEST-DRIVEN QUALITY ASSESSMENT IN OTHER DOMAINS In this part we showcase how the *Test-Driven Quality Assessment Methodology* can be directly applicable in other domains. In Chapter 12, *Assessing and Refining Mappings to RDF to Improve Dataset Quality*, we apply our methodology directly on the RDF mappings to identify quality issues. When a mapping is defined wrongly it can lead in multiple recurring data violations. We show that identifying such errors directly in the mappings is much faster and more efficient. We evaluate this setting on the DBpedia mappings as well as five other mapping datasets and discuss the results.

Chapter 13, *Semantically Enhanced Quality Assurance: The JURION Business Use Case*, applies our methodology in a real production system of Wolters Kluwers Deutschland (WKD). In particular, RDFUnit is used in a Continuous Integration (CI) system to verify the correct conversion of data and metadata to RDF. The results of this work are evaluated by WKD software developers and domain experts.

PART V - CONCLUSIONS AND FUTURE WORK Chapter 14 concludes and gives an outlook on future work. In addition to the contributions summarized above, it is worth noting that preliminary results of this thesis formed the *Data Quality* work package in the ALIGNED H2020 project. Moreover, 4 conference workshops and a special issue in Semantic Web Journal about data quality and DBpedia were co-organized by the author. Finally, this thesis resulted in the influence and authorship of SHACL, an upcoming W3C standard on RDF data Validation.

CONVENTIONS

Throughout this thesis, the namespace prefixes of Table 1 are used in turtle and SPARQL listings.

Prefix	*Namespace*
dbo	http://dbpedia.org/ontology/
dbp	http://dbpedia.org/property/
dbr	http://dbpedia.org/resource/
dbr-de	http://de.dbpedia.org/resource/
db-com	http://commons.dbpedia.org/resource/File:
dcterms	http://purl.org/dc/terms/
dc	http://purl.org/dc/elements/1.1/
commons-path	http://commons.wikimedia.org/wiki/Special:FilePath/
dc	http://purl.org/dc/elements/1.1/
foaf	http://xmlns.com/foaf/0.1/
geo	http://www.w3.org/2003/01/geo/wgs84_pos#
georss	http://www.georss.org/georss/
ls	http://spotlight.dbpedia.org/scores/
lx	http://spotlight.dbpedia.org/lexicalizations/
owl	http://www.w3.org/2002/07/owl#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
skos	http://www.w3.org/2004/02/skos/core#
sptl	http://spotlight.dbpedia.org/vocab/
wkdt	http://wikidata.org/entity/
xsd	http://www.w3.org/2001/XMLSchema#
void	http://rdfs.org/ns/void#

Table 1: List of namespace prefixes.

2

PRELIMINARIES

SEMANTIC WEB

The Semantic Web is a technology and specification stack that provides "common formats for integration and combination of data drawn from diverse sources" as well as a "language for recording how the data relates to real world objects". [1]

The term *Semantic web* was introduced by Tim Berners-Lee in 2001. *Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation. The first steps in weaving the Semantic Web into the structure of the existing Web are already under way. In the near future, these developments will usher in significant new functionality as machines become much better able to process and "understand" the data that they merely display at present* (Berners-Lee, Hendler, and Lassila, 2001b).

In the following, we describe the core building blocks of the Semantic Web

URIs & IRIs

A Uniform Resource Identifier (URI) (Berners-Lee, Fielding, and Masinter, 2005) is defined as a compact sequence of characters that identifies an abstract or physical resource. The URI syntax defines a grammar that is a superset of all valid URIs, allowing an implementation to parse the common components of a URI reference without knowing the scheme-specific requirements of every possible identifier. As depicted in Figure 1, a URI consists of five parts, namely the scheme, authority, path, query and fragment.

According to the URI syntax, only the US-ASCII coded character set is allowed and percent-encoded octets can be used to represent characters outside that range. This makes URIs with unicode characters not human-friendly

The Internationalized Resource Identifier (IRI) (Duerst and Suignard, 2005) is a specification that complements the Uniform Resource Identifier (URI). An IRI is defined as a sequence of characters from the Universal Character Set (Unicode/ISO 10646). The IRI specification defines a mapping from IRIs to URIs and thus, an IRIs can be used instead of URIs, where appropriate, to identify resources.

1 https://www.w3.org/2001/sw/ accessed 08/03/2017.

```
  foo://example.com:8042/over/there?name=ferret#nose
  \_/   \______________/\_________/ \_________/ \__/
   |           |            |            |        |
scheme     authority       path        query   fragment
   |   _____________________|__
  / \ /                        \
  urn:example:animal:ferret:nose
```

Figure 1: example URIs and their component parts (Berners-Lee, Fielding, and Masinter, 2005)

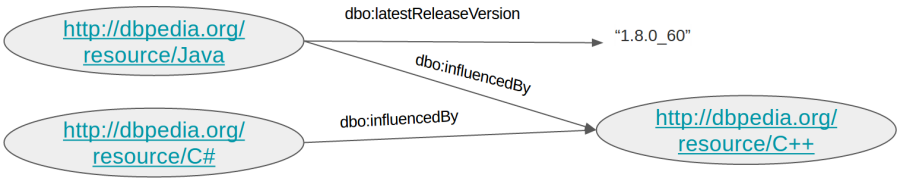


Figure 2: Example RDF Graph that consists of three triples

RDF

One of the foundation technologies of Semantic Web is the the Resource Description Framework (RDF). RDF is a framework for representing information in the Web and is defined by the RDF 1.1 Concept & Abstract Syntax (Cyganiak, Wood, and Lanthaler, 2014) and Semantics (Hayes and Patel-Schneider, 2014). The core part of RDF is a *triple* and a set of RDF triples constitute an RDF graph. A set of RDF graphs constitutes an RDF dataset.

An RDF triple consists of a subject, a predicate and an object. The subject can be an IRI or a blank node, the predicate can be an IRI and the object can be an IRI, a blank node or a literal. IRIs, as described in the previous section, are used to identify an abstract or a physical resource. Literals are used for values such as strings, numbers and dates and consist of two or three elements: the lexical form of the literal, the literal datatype and for language-tagged literals, a non empty language tag. Finally, blank nodes are identifiers that are locally scoped and are not persistent ot portable.

RDF defines an abstract data model that can have many different serializations such as N-Triples, N-Quads, Turtle, JSON-LD, TriG and XML. [2] Figure 2 depicts an example RDF graph that consists of three triples. The following example provides a serialization of that data model in turtle:

```
1 dbr:Java
2   dbo:latestReleaseVersion "1.8.0_60";
3   dbo:influencedBy dbr:C++ .
4 dbr:C#
5   dbo:influencedBy dbr:C++ .
```

2 https://www.w3.org/2011/rdf-wg/

RDF Schema languages: RDFS & OWL

There are two main W3C recommendations that serve as a schema language on top of RDF: RDF Schema (RDFS) (Guha and Brickley, 2014) and the Web Ontology Language (OWL) (Hitzler et al., 2009). Both languages are *monotonic*, meaning that entailments which hold before the addition of new information, also hold after it.

RDF Schema

RDFS defines a set classes with certain properties that provide the building blocks to describe ontologies or RDF vocabularies.

The main classes defined in RDFS are:

- **rdfs:Resource**: the class of everything
- **rdfs:Class**: the class of all classes
- **rdfs:Literal**: the class of all literal values
- **rdfs:Datatype**: the class of all datatypes
- **rdf:Property**: the class of all properties

The main properties defined in RDFS are `rdfs:label`, `rdfs:comment`, `rdfs:domain`, `rdfs:range`, `rdf:type`, `rdfs:subClassOf` and `rdfs:subPropertyOf`.

The following example defines a simple RDF vocabulary followed by instance data that use that vocabulary.

```
# RDF Schema
dbo:Person a rdfs:Class ;
  rdfs:label "Person" ;
  rdfs:comment "A humen being" .

dbo:birthDate a rdf:Property ;
  rdfs:label "birth date" ;
  rdfs:comment "The birthdate of a person" ;
  rdfs:domain dbo:Person ;
  rdfs:range xsd:date .

# instance data
ex:John a dbo:Person ;
  dbo:birthDate "1980-01-01"^^xsd:date .
```

Web Ontology Language

OWL is an extension of the semantics of RDFS, providing a richer and more expressive vocabulary. OWL is a *a powerful general-purpose modeling language for certain parts of human knowledge* that is designed to *formulate, exchange and reason with knowledge about a domain of interest*. The basic notions of OWL are

- **Axioms**: basic statements

- **Entities**: refer to real-world objects
- **Expressions**: combination of entities to create complex descriptions from simple ones.

OWL provides multiple profiles such as one based on Description Logic (OWL DL) or on Rule Languages (OWL2 RL). Profiles allow for different types of reasoning over RDF data, adhering to ontologies under such regimes.

Open & Closed World Assumption

Both RDF and OWL use the *Open World Assumption* (OWA). In OWA, the truth value of a statement may be true irrespective of whether or not it is known to be true. The OWA is the opposite of the *Closed World Assumption* (CWA) which is extensively used for validating RDF data.

Another major difference between OWA and CWA is how missing information is treated. This makes OWA unsuitable for validation of e.g. cardinality constraints.

```
Statement: Mary is born in Germany
Question: Is John born in Germany?

Answer in CWA: No
Answer in OWA: Unknown
```

Listing 1: OWA vs CWA when information is missing

It is worth noting that both schema languages were designed with inference as a primary use case and not validation and syntax conformance. Unlike for example XML, RDFS and OWL does not provide elaborate means to prescribe how a document should be structured. In particular, there is no way to enforce that a certain piece of information has to be present.

SPARQL Query Language

SPARQL (Prud'hommeaux, Harris, and Seaborne, 2013) is an RDF query language. SPARQL contains capabilities for querying required and optional graph patterns, as well as their conjunctions and disjunctions. SPARQL also supports query aggregations, subqueries, negation, and constraining queries by source RDF graph. The result of a SPARQL query can be a result sets or an RDF graph.

Triple patterns, also called as basic graph pattern are one of the most common parts of most SPARQL queries. Like RDF triples, triple patters also have a subject, a predicate and an object but each of them can be a variable. The following example triple pattern can be used to query the example RDF graph of Section 2.1.2

```
PREFIX dbr: <http://dbpedia.org/resource/>
```

```
2 PREFIX dbo: <http://dbpedia.org/ontyology/>
3 SELECT ?version WHERE {
4   dbr:Java dbo:latestReleaseVErsion ?version
5 }
```

A basic graph pattern matches a subgraph of the RDF data when RDF terms from that subgraph may be substituted for the variables and the result is RDF graph equivalent to the subgraph. For example, the result for the former SPARQL query would be *"1.8.0_60"*

In addition to triple patterns, SPARQL provides a big range of constructs that can be used to formulate complex queries on RDF graphs. These constructs include FILTERs, UNIONs, OPTIONALs and a big range of native functions. A complete list is available in Prud'hommeaux, Harris, and Seaborne, (2013)

Linked Data

"*The term Linked Data refers to a set of best practices for publishing and connecting structured data on the Web*" (Berners-Lee, Hendler, and Lassila, 2001a). By using the Linked Data (LD) best practices, structured data can be consumed by both humans and machines, have an explicitly defined meaning, have the ability to link to other data sources and to be linked from other data sources.

Although LD does not promote specific technologies, RDF is a perfectly fitting data model. Apart from RDF, LD utilize the HTTP protocol (RFC 2616 by Fielding et al., (1999)), the Uniform Resource Identifier (RFC 3986 by Berners-Lee, Fielding, and Masinter, (2005)) and the Transparent Content Negotiations (TCN) rules (RFC 2295 by Holtman and Mutz, (1998).

According to LD, structured data, are identified by URIs, i.e. using the http:// scheme. This way, the data can be easily accessed by dereferencing its URI through the HTTP protocol. The TCN rules are used to serve human or machine readable content depending on the agent (cf. Figure 3). In Sauermann and Cyganiak, (2008, Section 4.4), the authors suggest different strategies for dereferencing entities depending on the dataset size, entity count and application requirements. Section 4.4.1 proposes a new approach for dereferencing, when dealing with International Resource Identifiers (RFC 3987) (Duerst and Suignard, 2005).

Linking Open Data Project

The W3C SWEO community project[3] Linking Open Data (LOD) was initiated in October 2007 and aimed at the existence of large amounts of meaningfully interlinked Linked Data on the Web. Ever since, the

3 http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

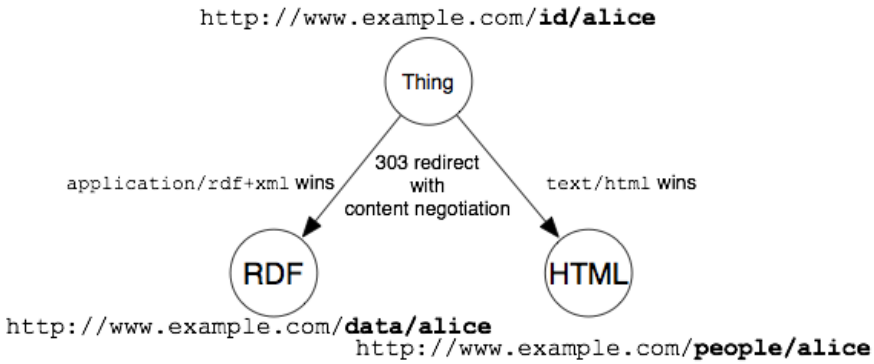


Figure 3: Entity content negotiation using 303 redirects (Sauermann and Cyganiak, 2008)

LOD project has published a big number of open datasets available on the Web according to the Linked Data principles and developed automated mechanisms to interlink them. All the participating datasets however had to meet certain criteria in order to be included in the LOD project, namely: 1) all items of interest should be identified using URI references, 2) all URI references should be resolvable on the Web to RDF descriptions, 3) every RDF triple with IRI as an object is conceived as a hyper link that can be followed by Semantic Web browsers and crawlers and 4) provide metadata about published data, so that clients can assess the quality of published data and choose between different means of access.

Figure 4 shows the data sets that have been published and interlinked by the LOD project so far. Collectively, the 570 data sets consist of over 31 billion RDF triples, which are interlinked by around 504 million RDF links (April 2014).[4]

The arcs in Figure 4 indicate that links exist between items in the two connected data sets. Heavier arcs roughly correspond to a greater number of links between two data sets, while bidirectional arcs indicate the outward links to the other exist in each data set. Certain data sets serve as linking hubs in the Web of Data. For example, the DBpedia data set consists of RDF triples extracted from Wikipedia pages. As DBpedia provides URIs and RDF descriptions for many common entities or concepts, it is frequently referenced in other more specialised data sets and have therefore developed into a hub to which an increasing number of other data sets is connected.

INFORMATION & DATA QUALITY

Information has, nowdays, become a valuable asset for any decision making mechanism. This need drove the creation of new research

4 Linking Open Data cloud diagram 2014, by Max Schmachtenberg, Christian Bizer, Anja Jentzsch and Richard Cyganiak. http://lod-cloud.net/

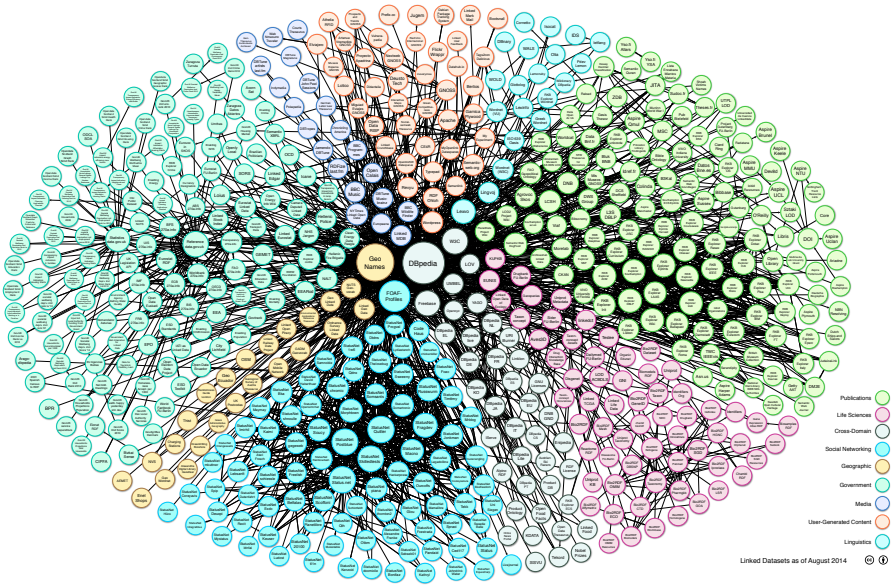


Figure 4: The 2014 LOD-Cloud diagram (image by Max Schmachtenberg, Christian Bizer, Anja Jentzsch and Richard Cyganiak. http://lod-cloud.net/)

areas and industry tools on making information available, reliable, accurate and keeping the information up-to-date. According to the latest ISO/IEC on *Information technology - Vocabulary* (ISO/IEC, 2015) and the *International Association for Information and Data Quality*[5] we have the following unified and merged definitions:

INFORMATION (1) knowledge concerning objects, such as facts, events, things, processes, or ideas, including concepts, that within a certain context has a particular meaning; (2) knowledge which reduces or removes uncertainty about the occurrence of a specific event from a given set of possible events; (3) Data in context, i.e., the meaning given to data or the interpretation of data based on its context; (4) the finished product as a result of processing, presentation and interpretation of data.

KNOWLEDGE Information context; understanding of the significance of information.

DATA (1) reinterpretable representation of information in a formalized manner suitable for communication, interpretation, or processing; (2) Symbols, numbers or other representation of facts; (3) The raw material from which information is produced when it is put in a context that gives it meaning; (4) raw, unrelated numbers or entries, e.g., in a database; raw forms of transactional representations.

5 http://iaidq.org/main/glossary.shtml

INFORMATION QUALITY (1) the fitness for use of information; information that meets the requirements of its authors, users, and administrators; (2) Consistently meeting all knowledge worker and end-customer expectations in all quality characteristics of the information products and services required to accomplish the enterprise mission (internal knowledge worker) or personal objectives (end customer); (3) The degree to which information consistently meets the requirements and expectations of all knowledge workers who require it to perform their processes.

INFORMATION QUALITY CHARACTERISTIC An aspect or property of information or information service that an information customer deems important in order to be considered "quality information". Characteristics include completeness, accuracy, timeliness, understandability, objectivity and presentation clarity, among others. Also called information quality "dimension".

INFORMATION QUALITY ASSESSMENT The random sampling of a data collection and measuring it against various quality characteristics, such as accuracy, completeness, validity, non-duplication or timeliness to determine its level of quality or reliability. Also called data quality assessment or data audit.

Using these official definitions, we see that data is just a formalization of information and information quality is defined as "fitness for use" of information. To assess the quality of information, one has to measure the information against some data quality dimensions. According to Bizer, (2007), these dimensions are grouped into four main categories:

- *Intrinsic*: dimensions that are independent of the user's context.
- *Contextual*: dimensions that are dependent of the user's context.
- *Representational*: dimensions that relate to the way information is represented in information systems.
- *Accessibility*: dimensions that relate to ways information is accessed.

Bizer defines 17 dimensions. Recent work in (Zaveri et al., 2015), extends the dimensions to 18 and additionally defines 69 metrics for measuring them.

Data quality in the context of RDF is thoroughly discussed in Chapter 11.

Part II

LARGE-SCALE MULTILINGUAL KNOWLEDGE EXTRACTION & PUBLISHING: THE CASE OF DBPEDIA

3

DBPEDIA OVERVIEW

Kontokostas and Hellmann, (2014) and Lehmann, Isele, Jakob, Jentzsch, Kontokostas, Mendes, Hellmann, Morsey, Kleef, Auer, and Bizer, (2015)

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 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 miles, or all Italian composers that were born in the 18th century.

The DBpedia project builds a large-scale, multilingual knowledge base by extracting structured data from Wikipedia editions in 130 languages. This knowledge base can be used to answer expressive queries such as the ones outlined above. Being multilingual and covering a 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 testbed in the research community and numerous applications, algorithms and tools have been built around or applied to DBpedia. 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 has developed into 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 – which will be explained in detail in Section 3.2 – 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.

1 http://www.alexa.com/topsites. Retrieved in March 2017.
2 http://meta.wikimedia.org/wiki/List_of_Wikipedias

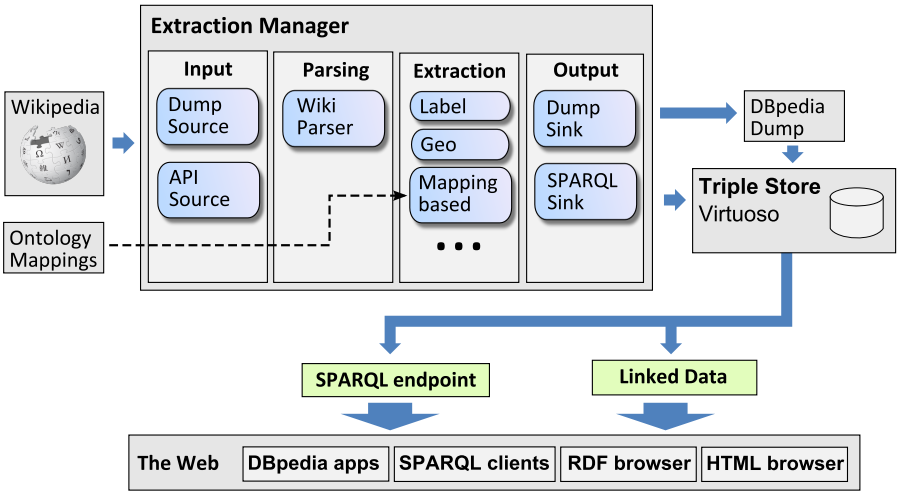


Figure 5: Overview of DBpedia extraction framework.

This chapter is structured as follows: In the next section (Section 3.1), we describe the DBpedia extraction framework, which forms the technical core of DBpedia. This is followed by an explanation of the community-curated DBpedia ontology with a focus on multilingual support (Section 3.2). Section 3.3 provides a list of DBpedia related projects that were co-developed as part of this thesis. Finally, Section 3.4 concludes and gives an outlook on the further development of DBpedia.

EXTRACTION FRAMEWORK

Wikipedia articles consist mostly of free text, but also comprise of various types of structured information in the form of wiki markup. Such information includes infobox templates, categorisation information, images, geo-coordinates, links to external web pages, disambiguation pages, redirects between pages, and links across different language editions of Wikipedia. The DBpedia extraction framework extracts this structured information from Wikipedia and turns it into a rich knowledge base. In this section, we give an overview of the DBpedia knowledge extraction framework.

General Architecture

Figure 5 shows an overview of the technical framework. The DBpedia extraction is structured into four phases:

INPUT: Wikipedia pages are read from an external source. Pages can either be read from a Wikipedia dump or directly fetched from a MediaWiki installation using the MediaWiki API.

PARSING: Each Wikipedia page is parsed by the wiki parser. The wiki parser transforms the source code of a Wikipedia page into an Abstract Syntax Tree.

EXTRACTION: The Abstract Syntax Tree of each Wikipedia page is forwarded to the extractors. DBpedia offers extractors for many different purposes, for instance, to extract labels, abstracts or geographical coordinates. Each extractor consumes an Abstract Syntax Tree and yields a set of RDF statements.

OUTPUT: The collected RDF statements are written to a sink. Different formats, such as N-Triples and Turtle, are supported.

Extractors

The DBpedia extraction framework employs various extractors for translating different parts of Wikipedia pages to RDF statements. A list of all available extractors is shown in Table 3. DBpedia extractors can be divided into four categories:

RAW INFOBOX EXTRACTION: The *raw infobox extraction* provides a direct mapping from infoboxes in Wikipedia to RDF. As the raw infobox extraction does not rely on explicit extraction knowledge in the form of mappings, the quality of the extracted data is lower. The raw infobox data is useful if a specific infobox has not been mapped yet and thus is not available in the mapping-based extraction.

MAPPING-BASED INFOBOX EXTRACTION: The *mapping-based infobox extraction* uses manually written mappings that relate infoboxes in Wikipedia to terms in the DBpedia ontology. The mappings also specify a datatype for each infobox property and thus help the extraction framework to produce high quality data. The mapping-based extraction will be described in detail in Section 3.1.4.

FEATURE EXTRACTION: The *feature extraction* uses a number of extractors that are specialized in extracting a single feature from an article, such as a label or geographic coordinates.

STATISTICAL EXTRACTION: Some NLP related extractors aggregate data from all Wikipedia pages in order to provide data that is based on *statistical measures* of page links or word counts.

Raw Infobox Extraction

The type of Wikipedia content that is most valuable for the DBpedia extraction are infoboxes. Infoboxes are frequently used to list an article's most relevant facts as a table of attribute-value pairs on the top

Name	*Description*	*Example*
abstract	Extracts the first lines of the Wikipedia article.	`dbr:Berlin dbo:abstract "Berlin is the capital city of (...)" .`
article categories	Extracts the categorization of the article.	`dbr:Oliver_Twist dc:subject dbr:Category:English_novels .`
category label	Extracts labels for categories.	`dbr:Category:English_novels rdfs:label "English novels" .`
category hierarchy	Extracts information about which concept is a category and how categories are related using the SKOS Vocabulary.	`dbr:Category:World_War_II skos:broader dbr:Category:Modern_history .`
disambiguation	Extracts disambiguation links.	`dbr:Alien dbo:wikiPageDisambiguates dbr:Alien_(film) .`
external links	Extracts links to external web pages related to the concept.	`dbr:Animal_Farm dbo:wikiPageExternalLink <http://books.google.com/?id=RBGmrDnBs8UC> .`
geo coordinates	Extracts geo-coordinates.	`dbr:Berlin georss:point "52.5006 13.3989" .`
grammatical gender	Extracts grammatical genders for persons.	`dbr:Abraham_Lincoln foaf:gender "male" .`
homepage	Extracts links to the official homepage of an instance.	`dbr:Alabama foaf:homepage <http://alabama.gov/> .`
image	Extracts the first image of a Wikipedia page.	`dbr:Berlin foaf:depiction <http://.../Overview_Berlin.jpg> .`
infobox	Extracts all properties from all infoboxes.	`dbr:Animal_Farm dbo:date "March 2010" .`
interlanguage	Extracts interwiki links.	`dbr:Albedo dbo:wikiPageInterLanguageLink dbr-de:Albedo .`
label	Extracts the article title as label.	`dbr:Berlin rdfs:label "Berlin" .`
lexicalizations	Extracts information about surface forms and their association with concepts (only N-Quad format).	`dbr:Pine sptl:lexicalization lx:pine_tree ls:Pine_pine_tree . lx:pine_tree rdfs:label "pine tree" . ls:Pine_pine_tree sptl:pUriGivenSf "0.941" .`
mappings	Extraction based on mappings of Wikipedia infoboxes to the DBpedia ontology.	`dbr:Berlin dbo:country dbr:Germany .`
page ID	Extracts page ids of articles.	`dbr:Autism dbo:wikiPageID "25" .`
page links	Extracts all links between Wikipedia articles.	`dbr:Autism dbo:wikiPageWikiLink dbr:Human_brain .`
persondata	Extracts information about persons represented using the PersonData template.	`dbr:Andre_Agassi foaf:birthDate "1970-04-29" .`
PND	Extracts PND (Personennamendatei) data about a person.	`dbr:William_Shakespeare dbo:individualisedPnd "118613723" .`
redirects	Extracts redirect links between articles in Wikipedia.	`dbr:ArtificialLanguages dbo:wikiPageRedirects dbr:Constructed_language .`
revision ID	Extracts the revision ID of the Wikipedia article.	`dbr:Autism prov:wasDerivedFrom enwiki:Autism?oldid=495234324 .`
thematic concept	Extracts 'thematic' concepts, the centres of discussion for categories.	`dbr:Category:Music skos:subject dbr:Music .`
topic signatures	Extracts topic signatures.	`dbr:Alkane sptl:topicSignature "carbon alkanes atoms" .`
wiki page	Extracts links to corresponding articles in Wikipedia.	`dbr:AnAmericanInParis foaf:isPrimaryTopicOf enwiki:AnAmericanInParis .`

Table 3: Overview of the DBpedia extractors.

right-hand side of the Wikipedia page (for right-to-left languages on the top left-hand side). Infoboxes that appear in a Wikipedia article are based on a template that specifies a list of attributes that can form the infobox. A wide range of infobox templates are used in Wikipedia. Common examples are templates for infoboxes that describe persons, organisations or automobiles. As Wikipedia's infobox template system has evolved over time, different communities of Wikipedia editors use different templates to describe the same type of things (e.g. `Infobox_city_japan`, `Infobox_swiss_town` and `Infobox_town_de`). In addition, different templates use different names for the same attribute (e.g. `birthplace` and `placeofbirth`). As many Wikipedia editors do not strictly follow the recommendations given on the page that describes a template, attribute values are expressed using a wide range of different formats and units of measurement. An excerpt of an infobox that is based on a template for describing automobiles is shown below:

```
{{Infobox automobile
| name         = Ford GT40
| manufacturer = [[Ford Advanced Vehicles]]
| production   = 1964-1969
| engine       = 4181cc
(...)
}}
```

In this infobox, the first line specifies the infobox type and the subsequent lines specify various attributes of the described entity.

An excerpt of the extracted data is as follows:

```
dbr:Ford_GT40
  dbp:name      "Ford GT40"@en;
  dbp:manufacturer dbr:Ford_Advanced_Vehicles;
  dbp:engine    4181;
  dbp:production 1964;
  (...).
```

This extraction output has weaknesses: The resource is not associated to a class in the ontology and parsed values are cleaned up and assigned a datatyper based on heuristics. In particular, the raw infobox extractor searches for values in the following order: dates, coordinates, numbers, links and strings as default. Thus, the datatype assignment for the same property in different resources is non deterministic. The `engine` for example is extracted as a number but if another instance of the template used "cc4181" it would be extracted as string. This behaviour makes querying for properties in the `dbp` namespace inconsistent. Those problems can be overcome by the mapping-based infobox extraction presented in the next subsection.

Mapping-Based Infobox Extraction

In order to homogenize the description of information in the knowledge base, in 2010 a community effort was initiated to develop an ontology schema and mappings from Wikipedia infobox properties to this ontology. The alignment between Wikipedia infoboxes and the ontology is performed via community-provided mappings that help to normalize name variations in properties and classes. Heterogeneity in the Wikipedia infobox system, like using different infoboxes for the same type of entity or using different property names for the same property (cf. Section 3.1.3), can be alleviated in this way. This significantly increases the quality of the raw Wikipedia infobox data by typing resources, merging name variations and assigning specific datatypes to the values.

This effort is realized using the DBpedia Mappings Wiki[3], a MediaWiki installation set up to enable users to collaboratively create and edit mappings. These mappings are specified using the DBpedia Mapping Language. The mapping language makes use of MediaWiki templates that define DBpedia ontology classes and properties as well as template/table to ontology mappings. A mapping assigns a type from the DBpedia ontology to the entities that are described by the corresponding infobox. In addition, attributes in the infobox are mapped to properties in the DBpedia ontology. In the following, we show a mapping that maps infoboxes that use the `Infobox_automobile` template to the DBpedia ontology:

```
{{TemplateMapping
|mapToClass = Automobile
|mappings =
 {{PropertyMapping
 | templateProperty = name
 | ontologyProperty = foaf:name }}
 {{PropertyMapping
 | templateProperty = manufacturer
 | ontologyProperty = manufacturer }}
 {{DateIntervalMapping
 | templateProperty = production
 | startDateOntologyProperty =
     productionStartDate
 | endDateOntologyProperty =
     productionEndDate }}
 {{IntermediateNodeMapping
 | nodeClass = AutomobileEngine
 | correspondingProperty = engine
 | mappings =
    {{PropertyMapping
    | templateProperty = engine
    | ontologyProperty = displacement
    | unit = Volume }}
    {{PropertyMapping
```

3 http://mappings.dbpedia.org

```
    | templateProperty = engine
    | ontologyProperty = powerOutput
    | unit = Power }}
 }}
 (...)
}}
```

The RDF statements that are extracted from the previous infobox example are shown below. As we can see, the production period is correctly split into a start year and an end year and the engine is represented by a distinct RDF node. It is worth mentioning that all values are canonicalized to basic units. For example, in the `engine` mapping we state that `engine` is a *Volume* and thus, the extractor converts "4181cc" (cubic centimeters) to cubic meters ("0.004181"). Additionally, there can exist multiple mappings on the same property that search for different datatypes or different units. For example, a number with "PS" as a suffix for `engine`.

```
dbr:Ford_GT40
  rdf:type   dbo:Automobile;
  rdfs:label "Ford GT40"@en;
  dbo:manufacturer  dbr:Ford_Advanced_Vehicles;
  dbo:productionStartYear   "1964"^^xsd:gYear;
  dbo:productionEndYear  "1969"^^xsd:gYear;
  dbo:engine [
             rdf:type AutomobileEngine;
             dbo:displacement "0.004181";
  ]
  (...) .
```

The DBpedia Mapping Wiki is not only used to map different templates within a single language edition of Wikipedia to the DBpedia ontology, but is used to map templates from all Wikipedia language editions to the shared DBpedia ontology. Figure 6 shows how the infobox properties *author* and συγγραφεασ – author in Greek – are both being mapped to the global identifier `dbo:author`. That means, in turn, that information from all language versions of DBpedia can be merged and DBpedias for smaller languages can be augmented with knowledge from larger DBpedias such as the English edition. Conversely, the larger DBpedia editions can benefit from more specialized knowledge from localized editions, such as data about smaller towns which is often only present in the corresponding language edition (Tacchini, Schultz, and Bizer, 2009).

Besides hosting of the mappings and DBpedia ontology definition, the DBpedia Mappings Wiki offers various tools which support users in their work:

- **Mapping Syntax Validator** The mapping syntax validator checks for syntactic correctness and highlights inconsistencies such as missing property definitions.

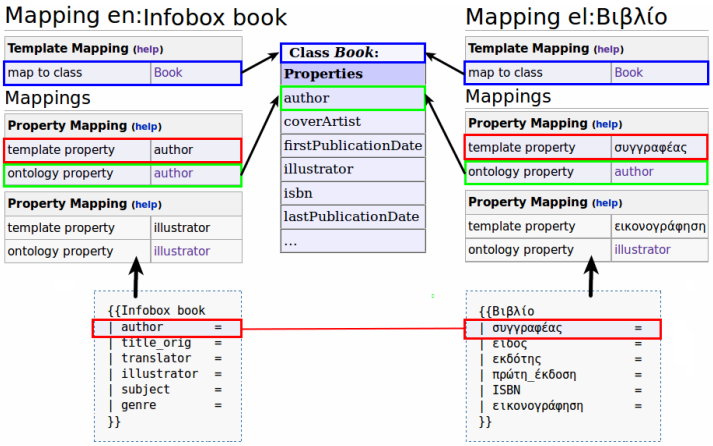


Figure 6: Depiction of the mapping from the Greek (left) and English Wikipedia templates (right) about books to the same DBpedia Ontology class (middle) (Kontokostas et al., 2012).

- **Extraction Tester** The extraction tester linked on each mapping page tests a mapping against a set of example Wikipedia pages. This gives direct feedback about whether a mapping works and how the resulting data is structured.
- **Mapping Tool** The DBpedia Mapping Tool is a graphical user interface that supports users to create and edit mappings.

URI Schemes

For every Wikipedia article, the framework introduces a number of URIs to represent the concepts described on a particular page. Up to 2011, DBpedia published URIs only under the `http://dbpedia.org` domain. The main namespaces were:

- `http://dbpedia.org/resource/` (prefix *dbr*) for representing article data. There is a one-to-one mapping between a Wikipedia page and a DBpedia resource based on the article title. For example, for the Wikipedia article on *Berlin*[4], DBpedia will produce the URI *dbr:Berlin*. Exceptions in this rule appear when intermediate nodes are extracted from the mapping-based infobox extractor as unique URIs (e.g., the `engine` mapping example in Section 3.1.4).
- `http://dbpedia.org/property/` (prefix *dbp*) for representing properties extracted from the raw infobox extraction (cf. Section 3.1.3), e.g. *dbp:population*.
- `http://dbpedia.org/ontology/` (prefix *dbo*) for representing the DBpedia ontology (cf. Section 3.1.4), e.g. *dbo:populationTotal*.

4 `http://en.wikipedia.org/wiki/Berlin`

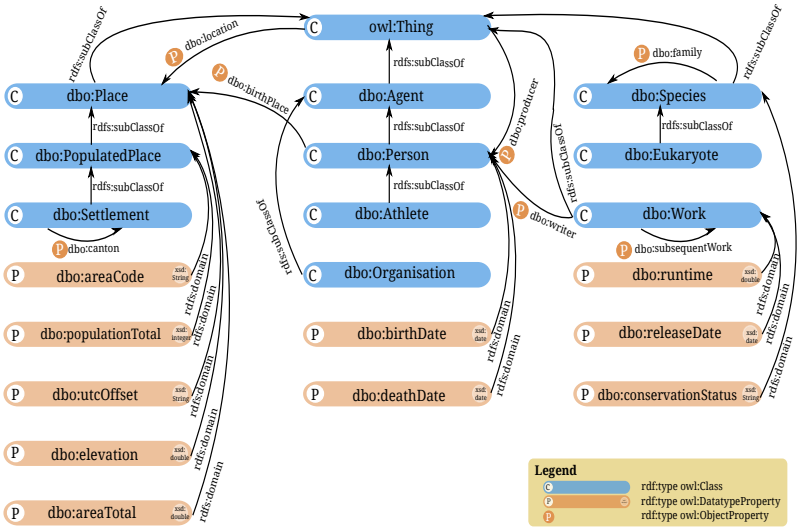


Figure 7: Snapshot of a part of the DBpedia ontology.

Although data from other Wikipedia language editions were extracted, they used the same namespaces. This was achieved by exploiting the Wikipedia *inter-language links*[5]. For every page in a language other than English, the page was extracted only if the page contained an inter-language link to an English page. In that case, using the English link, the data was extracted under the English resource name (i.e. *dbr:Berlin*).

Recent DBpedia internationalisation developments showed that this approach omitted valuable data (Kontokostas et al., 2012) (cf. Chapter 4). Thus, starting from the *DBpedia 3.7 release*[6], two types of data sets were generated. The *localized data sets* contain all things that are described in a specific language. Within the datasets, things are identified with language specific URIs such as `http://<lang>.dbpedia.org/resource/` for article data and `http://<lang>.dbpedia.org/property/` for property data. In addition, we produce a *canonicalized data set* for each language (see Section 4.3 for more details). The canonicalized data sets only contain things for which a corresponding page in the English edition of Wikipedia exists. Within all canonicalized data sets, the same thing is identified with the same URI from the generic language-agnostic namespace `http://dbpedia.org/resource/`.

DBPEDIA ONTOLOGY

The DBpedia ontology consists of 320 classes which form a subsumption hierarchy and are described by 1,650 different properties. With a maximal depth of 5, the subsumption hierarchy is intentionally kept rather shallow which fits use cases in which the ontology is visual-

5 http://en.wikipedia.org/wiki/Help:Interlanguage_links
6 A list of all DBpedia releases is provided in Table 4

ized or navigated. Figure 7 depicts a part of the DBpedia ontology, indicating the relations among the top ten classes of the DBpedia ontology, i.e. the classes with the highest number of instances. The complete DBpedia ontology can be browsed online at http://mappings.dbpedia.org/server/ontology/classes/.

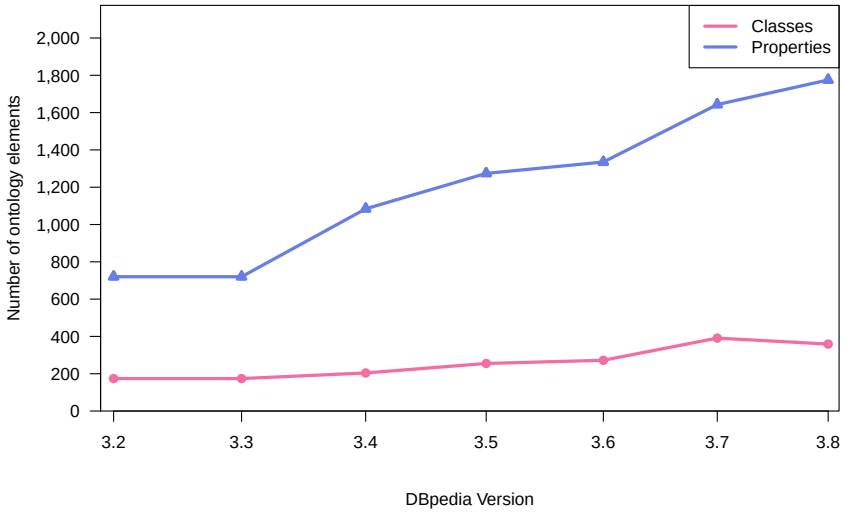


Figure 8: Growth of the DBpedia ontology.

The DBpedia ontology is maintained and extended by the community in the DBpedia Mappings Wiki. Figure 8 depicts the growth of the DBpedia ontology over time. While the number of classes is not growing too much due to the already good coverage of the initial version of the ontology, the number of properties increases over time due to the collaboration on the DBpedia Mappings Wiki and the addition of more detailed information to infoboxes by Wikipedia editors.

Mapping Statistics

As of April 2013, there exist mapping communities for 27 languages, 23 of which are active. Figure 9 shows statistics for the coverage of these mappings in DBpedia. Figures (a) and (c) refer to the absolute number of template and property mappings that are defined for every DBpedia language edition. Figures (b) and (d) depict the percentage of the defined template and property mappings compared to the total number of available templates and properties for every Wikipedia language edition. Figures (e) and (g) show the occurrences (instances) that the defined template and property mappings have in Wikipedia. Finally, figures (f) and (h) give the percentage of the mapped templates and properties occurences, compared to the total templates and property occurences in a Wikipedia language edition.

It can be observed in the figure that the Portuguese DBpedia language edition is the most complete regarding mapping coverage. Other

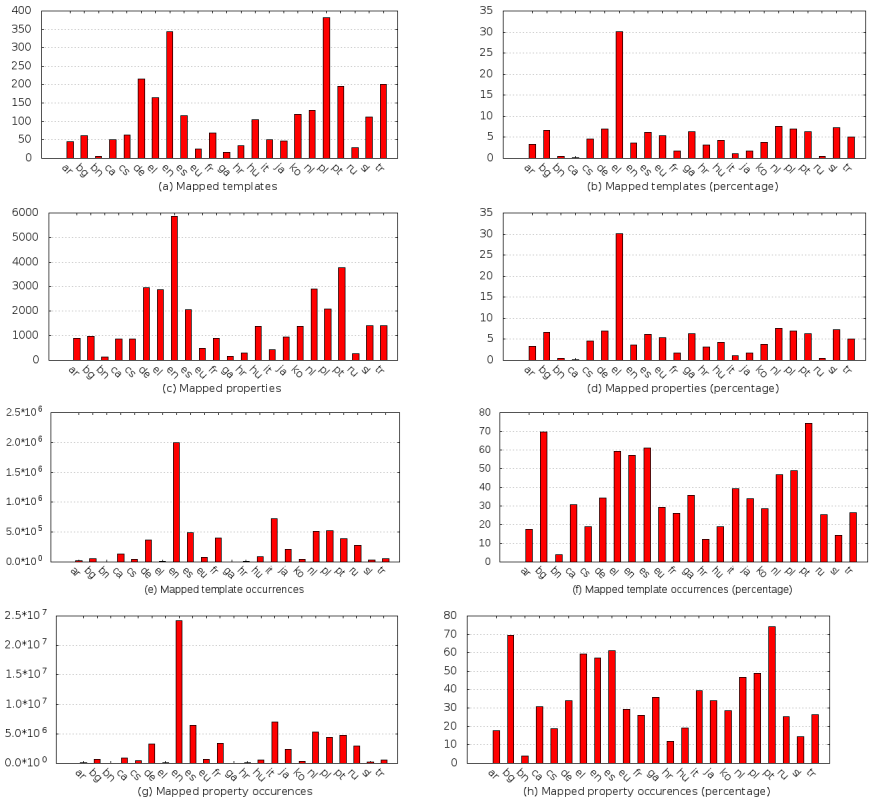


Figure 9: Mapping coverage statistics for all mapping-enabled languages.

language editions such as Bulgarian, Dutch, English, Greek, Polish and Spanish have mapped templates covering more than 50% of total template occurrences. In addition, almost all languages have covered more than 20% of property occurrences, with Bulgarian and Portuguese reaching up to 70%.

The mapping activity of the ontology enrichment process along with the editing of the ten most active mapping language communities is depicted in Figure 10. It is interesting to notice that the high mapping activity peaks coincide with the DBpedia release dates. For instance, the DBpedia 3.7 version was released on September 2011 and the 2nd and 3rd quarter of that year have a very high activity compared to the 4th quarter. In the last two years (2012 and 2013), most of the DBpedia mapping language communities have defined their own chapters and have their own release dates. Thus, recent mapping activity shows less fluctuation.

Finally, Figure 11 shows the English property mappings occurrence frequency. Both axes are in log scale and represent the number of property mappings (x axis) that have exactly y occurrences (y axis). The occurrence frequency follows a *long tail* distribution. Thus, a low number of property mappings have a high number of occurrences and a high number of property mappings have a low number of occurences.

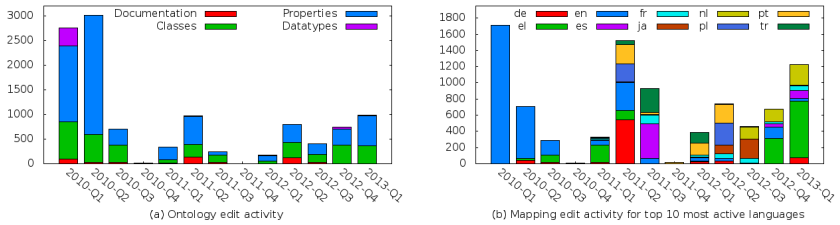


Figure 10: Mapping community activity for (a) ontology and (b) 10 most active language editions

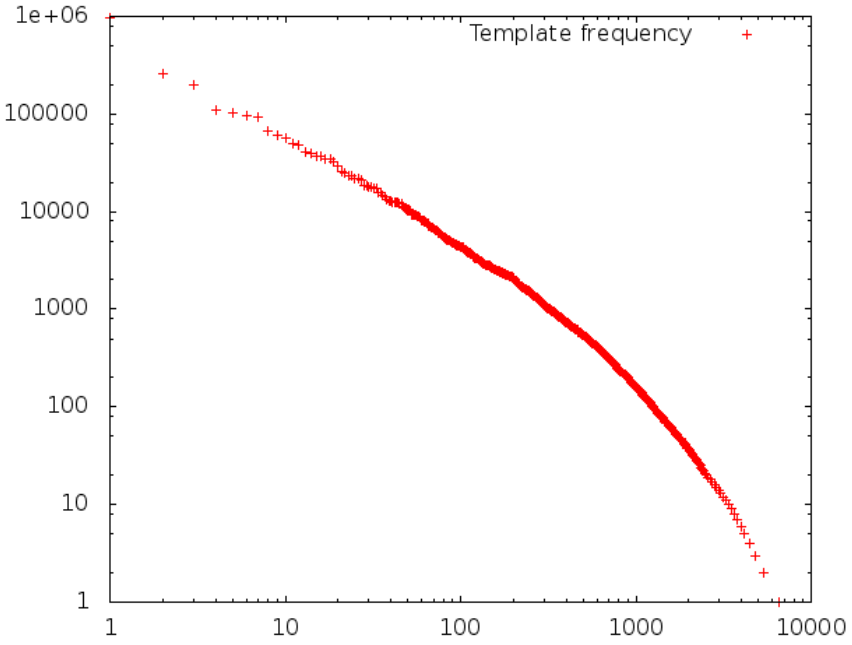


Figure 11: English property mappings occurrence frequency (both axes are in log scale)

Internationalisation Community

The introduction of the mapping-based infobox extractor alongside live synchronisation approaches in (Hellmann et al., 2009) allowed the international DBpedia community to easily define infobox-to-ontology mappings. As a result of this development, there are presently mappings for 27 languages[7]. The DBpedia 3.7 release[8] in September 2011 was the first DBpedia release to use the localized I18n (Internationalisation) DBpedia extraction framework (Kontokostas et al., 2012).

At the time of writing, DBpedia chapters for 14 languages have been founded: Basque, Czech, Dutch, English, French, German, Greek, Italian, Japanese, Korean, Polish, Portuguese, Russian and Spanish.[9] Besides providing mappings from infoboxes in the corresponding Wikipedia editions, DBpedia chapters organise a local community and provide hosting for data sets and associated services.

While at the moment chapters are defined by ownership of the IP and server of the sub domain A record (e.g. http://ko.dbpedia.org) given by DBpedia maintainers, the DBpedia internationalisation committee[10] is manifesting its structure and each language edition has a representative with a vote in elections. In some cases (e.g. Greek[11] and Dutch[12]) the existence of a local DBpedia chapter has had a positive effect on the creation of localized LOD clouds (Kontokostas et al., 2012).

In the weeks leading to a new release, the DBpedia project organises a *mapping sprint*, where communities from each language work together to improve mappings, increase coverage and detect bugs in the extraction process. The progress of the mapping effort is tracked through statistics on the number of mapped templates and properties, as well as the number of times these templates and properties occur in Wikipedia. These statistics provide an estimate of the coverage of each Wikipedia edition in terms of how many entities will be typed and how many properties from those entities will be extracted. Therefore, they can be used by each language edition to prioritize properties and templates with higher impact on the coverage.

The mapping statistics have also been used as a way to promote a healthy competition between language editions. A sprint page was created with bar charts that show how close each language is from

7 Arabic (ar), Bulgarian (bg), Bengali (bn), Catalan (ca), Czech (cs), German (de), Greek (el), English (en), Spanish (es), Estonian (et), Basque (eu), French (fr), Irish (ga), Hindi (hi), Croatian (hr), Hungarian (hu), Indonesian (id), Italian (it), Japanese (ja), Korean (ko), Dutch (nl), Polish (pl), Portuguese (pt), Russian (ru), Slovene (sl), Turkish (tr), Urdu (ur)

8 http://blog.dbpedia.org/2011/09/11/

9 Accessed on 25/09/2013: http://wiki.dbpedia.org/Internationalization/Chapters

10 http://wiki.dbpedia.org/Internationalization

11 http://el.dbpedia.org

12 http://nl.dbpedia.org

achieving total coverage (as shown in Figure 9), and line charts showing the progress over time highlighting when one language is overtaking another in their race for higher coverage. The mapping sprints have served as a great motivator for the crowd-sourcing efforts, as it can be noted from the increase in the number of mapping contributions in the weeks leading to a release.

SUMMARY OF OTHER RECENT DEVELOPMENTS

Besides the core DBpedia development, it is worth mentioning three additional related projects: DataID (Section 3.3.1), an extensible dataset metadata description ontology, DBTax (Section 3.3.2), a data-driven DBpedia taxonomy and, DBpedia Viewer (Section 3.3.3), an integrative interface for the DBpedia ecosystem.

In addition to research related projects, Table 4 provides an overview of the project's evolution through time.

DBpedia Metadata with DataID

Brümmer, Baron, Ermilov, Freudenberg, Kontokostas, and Hellmann, (2014) and Freudenberg, Brummer, Rucknagel, Ulrich, Eckart, Kontokostas, and Hellmann, (2016)

DataID provides semantically rich metadata for complex datasets. The DataID ontology was originally created to thoroughly describe a DBpedia release (Brümmer et al., 2014). However, since the DBpedia releases are quite complex, DataID is capable to describe any dataset.

Recent developments in DataID (Freudenberg et al., 2016), resulted in a more modular design that is aligned with current standards on dataset metadata like DCAT (*Data Catalog Vocabulary (DCAT)*) and VOID (*Describing Linked Datasets with the VoID Vocabulary*). The onion-like layer model illustrates the different DataID layers. An ontology of a certain layer shall only import DataID ontologies from layers below their own. The mid-layer (or common extensions) of this model is comprised of highly reusable ontologies, extending DataID core to cover additional aspects of dataset metadata. While non of them are a mandatory import for use case specific extensions, as opposed to DataID core, in many cases some or all of them will be useful contributions.

DataID core provides the basic description of a dataset and serves as foundation for all extensions to DataID.

Linked Data[13] extends DataID core with the VOID vocabulary (*Describing Linked Datasets with the VoID Vocabulary*) and some additional properties specific to LOD datasets. Many VOID and Linked Data references from the previous version of DataID were outsourced into this ontology.

13 https://github.com/dbpedia/DataId-Ontology/tree/master/ld

Table 4: DBpedia timeline.

Year	*Month*	*Event*
2006	Nov	Start of infobox extraction from Wikipedia
2007	Mar	DBpedia 1.0 release
	Jun	ESWC DBpedia article (Auer and Lehmann, 2007)
	Nov	DBpedia 2.0 release
	Nov	ISWC DBpedia article (Auer et al., 2008)
	Dec	DBpedia 3.0 release candidate
2008	Feb	DBpedia 3.0 release
	Aug	DBpedia 3.1 release
	Nov	DBpedia 3.2 release
2009	Jan	DBpedia 3.3 release
	Sep	JWS DBpedia article (Bizer et al., 2009)
2010	Feb	Information extraction framework in Scala
		Mappings Wiki release
	Mar	DBpedia 3.4 release
	Apr	DBpedia 3.5 release
	May	Start of DBpedia Internationalization effort
2011	Feb	DBpedia Spotlight release
	Mar	DBpedia 3.6 release
	Jul	DBpedia Live release
	Sep	DBpedia 3.7 release (with I18n datasets)
2012	Aug	DBpedia 3.8 release
	Sep	Publication of DBpedia Internationalization article (Kontokostas et al., 2012)
2013	Sep	DBpedia 3.9 release
2014	Aug	DBpedia Commons release (cf. Chapter 5)
	Sep	DBpedia 2014 release
2015	Jun	DBpedia Wikidata release (cf. Chapter 6)
	Aug	DBpedia 2015-04 release with DBpedia Commons (Vaidya et al., 2015), DBTax (Fossati, Kontokostas, and Lehmann, 2015) and DataID (Freudenberg et al., 2016)
2016	Mar	DBpedia 2015-10 release
2017	Jan	Publication of DBpedia Wikidata (Ismayilov et al., 2016) accepted

Activities & Plans[14] provides provenance information of activities which generated, changed or used datasets. The goal is to record all activities needed to replicate a dataset as described by a DataID. Plans can describe which steps (activities, precautionary measures) are put in place to reach a certain goal. This extension relies heavily on the PROV ontology (*The PROV Ontology*).

14 https://github.com/dbpedia/DataId-Ontology/tree/DataManagementPlanExtension/acp

Statistics will provide the necessary measures to publish multi-dimensional data, such as statistics about datasets, based on the Data Cube Vocabulary (*The RDF Data Cube Vocabulary*).

DataID is in use on DBpedia since 2015, describing the different DBpedia datasets (e.g. for the v2015-10 English DBpedia[15]). There is currently an effort on implementing a DataID service and website to simplify and automate the creation, validation and dissemination of DataIDs, supporting humans in creating DataIDs manually, as well as automation tasks with a service endpoint. Finally, DataID core is planned to be published as a W3C member submission.

Unsupervised Learning of an Extensive and Usable DBpedia Taxonomy

Fossati, Kontokostas, and Lehmann, (2015)

The DBpedia ontology (DBO) is highly heterogeneous in terms of granularity (cf. for instance the classes Band versus SambaSchool, both direct subclasses of Organisation) and is supposed to encapsulate the entire encyclopedic world. This indicates there is ample room to improve the quality of DBO. The Wikipedia category system is a fine-grained topical classification of Wikipedia articles, thus being natively suitable for encoding Wikipedia knowledge. DBpedia uses the category hierarchy as a supplementary classification system, while several taxonomization efforts such as (Flati et al., 2014; Hoffart et al., 2013b; Melo and Weikum, 2010; Nastase and Strube, 2013; Nastase et al., 2010; Ponzetto and Strube, 2007, 2011), aim at mapping categories into types. However, their granularity is often very high, resulting in an arguably overly large set of items. From a practical perspective, it is vital to cluster resources into classes with intuitive labels, in order to simplify the end user's cognitive effort needed when querying the knowledge base. Hence, identifying a taxonomy based on a prominent subset of Wikipedia categories is a critical step to both extend and homogenize DBPO.

Furthermore, a clear problem of coverage has been recently pointed out (Aprosio, Giuliano, and Lavelli, 2013; Gangemi et al., 2012; Paulheim and Bizer, 2013; Pohl, 2012). For instance, although the English Wikipedia contains about 4.9 million articles, DBpedia has only classified 2.8 million into DBPO. One of the major reasons is that a significant amount of Wikipedia entries does not contain an infobox, which is a valuable piece of information to infer the type of an entry. This results in a large number of untyped entities, thus restraining the exploitation of the knowledge base. Consequently, the extension of the DBpedia data coverage is a crucial step towards the release of richly structured and high quality data.

DBTax (Fossati, Kontokostas, and Lehmann, 2015) is a completely data-driven methodology to automatically construct a comprehensive classification of DBpedia resources. DBTax provides: an Exhaustive

15 http://downloads.dbpedia.org/2015-10/core-i18n/en/2015-10_dataid_en.ttl

type coverage over the whole knowledge base, focuses on the actual usability of the schema from an end user's perspective and, forms a fully unsupervised algorithm for building the taxonomy.

As described in (Fossati, Kontokostas, and Lehmann, 2015) in detail, DBTax is constructed in the following four major steps:

1. Leaf node extraction;
2. Prominent node discovery;
3. Class taxonomy generation (T-Box);
4. Pages type assignment (A-Box).

The result of this work is:

- The taxonomy (T-Box) automatically generated according to step 3 is composed of 1,902 classes;
- 10,729,507 *instance-of* assertions (A-Box) are produced as output of step 4. An example is reported as follows.

```
1 dbr:Combat_Rock a dbtax:Album .
```

- A total of 4,260,530 unique resources are assigned a type, 2,325,506 of which do not have one in the DBpedia 3.9 release.

DBTax is thoroughly evaluated against other ontologies and taxonomies providing excellent results.

DBpedia Viewer - An Integrative Interface of the DBpedia ecosystem

Lukovnikov, Stadler, Kontokostas, Hellmann, and Lehmann, (2014)

This section describes DBpedia Viewer, the new DBpedia interface, which aims to present information from DBpedia in an engaging way while adhering to the Linked Data principles (Lukovnikov et al., 2014). DBpedia Viewer integrates existing DBpedia services as well as external Linked Data visualization tools to improve user-friendliness. The DBpedia interface was originally envisioned to serve data from DBpedia datasets, but could lead to a customizable framework that can easily be configured for other datasets. With the new DBpedia interface, we aim to show that even a generic representation of RDF data can be user-friendly and offer relevant services. A distinguishing feature of DBpedia Viewer is the Triple Action Framework, a framework which allows to dynamically associate action types with triples.

Following we provide an elaborate discussion of the new DBpedia Viewer features as depicted in Figure 12.

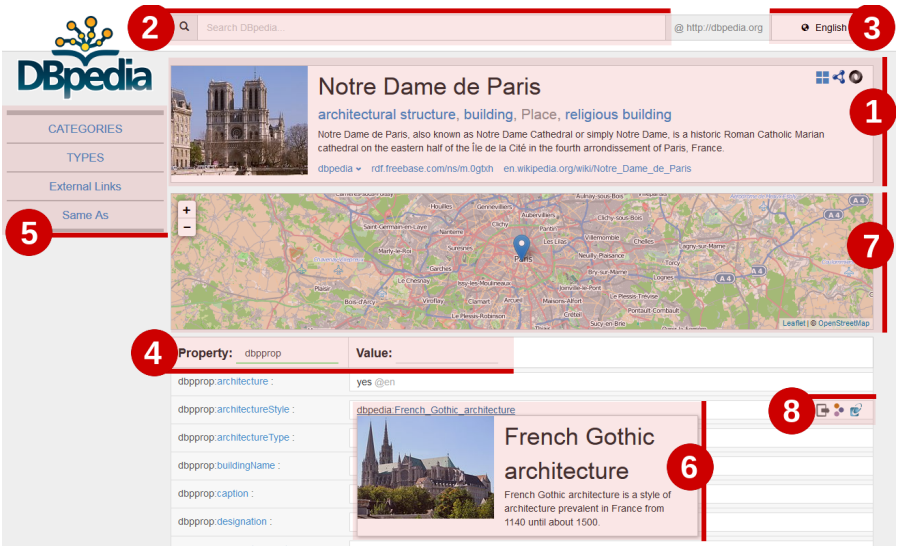


Figure 12: Screenshot of the new interface. The transparent red areas highlight the DBpedia viewer new features: (1) pretty box, (2) search bar, (3) language switcher, (4) triple filter, (5) shortcuts, (6) preview box, (7) map and (8) triple actions.

PRETTY BOX The *pretty box* (part one of Figure 12) displays important properties of the viewed entity. There is a predefined set of facts we provide, namely: (1) a picture, (2) the title, (3) the types, (4) a short description and (5) links to other resources. These data are generated from the set of triples describing the viewed resources using predefined mappings. The DBpedia datasets provide most of this general information for all entities. In some cases, however, the picture or links to other resources are not available. The DBpedia Viewer does not perform automatic selection of relevant properties to display. This is out of the scope of this project. Our goal was to develop a UI that is customizable to configure relevant properties used to adapt the view.

In the top right corner of the pretty box (Figure 12), three icons trigger the *entity actions*. The currently available actions are links to alternate data representations (XML/RDF, n3, JSON-LD, ...) as well as links to consult the resource using alternative Linked Data browsers.

SEARCH BAR DBpedia Viewer provides search functionality with auto-complete capabilities by re-using the DBpedia Lookup service. DBpedia Lookup enables searching for DBpedia entities using strings or provides prefix-based suggestions.

LANGUAGE FILTERING The language filtering system allows the user to choose a preferred display language. This filters all literal values based on the user preferences and displays only the relevant values. This feature is helpful on dbpedia.org where labels and abstracts exist in 12 different languages. In the case a literal does not

exist in the preferred language, a fallback language (usually English) is chosen by default.

TRIPLE FILTERING Part four of Figure 12 highlights the triple filtering feature. Triples can be filtered using both properties and values. This is useful for the users who quickly want to find specific properties and values. The filtering is based on string matching and supports all literal values as well as URIs.

SHORTCUT BOX The shortcut box (part 5 in Figure 12) provides anchor links to some important properties of entities. However, the list of properties is currently hardcoded and contains links to categories, types, external links, etc...

LIVE PREVIEWS When the user hovers over a DBpedia link (URI, ontology property or class) a concise, language-filtered preview is displayed. For entities, this preview contains a picture (if available), the title and a short description. Part 6 of Figure 12 shows a preview of the French Gothic architecture entity.

MAPS For entities having location information (latitude and longitude), a map is shown with its coordinates. OpenStreetMap is used for the map display.

TRIPLE ACTIONS As displayed in part 8 of (Figure 12), next to each triple, different icons exist, each representing a different *triple action*. Triple actions are enabled using conditions on the triple. Thus, the set of available actions for different triples may be different. When the conditions are met, the action icon is displayed next to the triple. When the user clicks on the triple action icon, the action is executed. Below is an overview of the currently implemented user actions:

- Annotation – uses DBpedia Spotlight to annotate text. Only applicable to texts of certain length.
- RelFinder – links to RelFinder, where the connections (including indirect ones) between the viewed entity and the value entity can be explored. Only applicable to DBpedia resources.
- LodLive – opens the value entity with the LodLive browser. Only applicable to DBpedia resources.
- OpenLink Faceted Browser – view the value entity using OpenLink Faceted Browser. Only applicable to DBpedia resources.
- Wikipedia – opens the Wikipedia page associated with the value entity. Only applicable to DBpedia resources.

- DBpedia template mapping – links to the DBpedia mapping associated with the DBpedia template. Only applicable to DBpedia resources under the Wikipedia template namespace.

CONCLUSIONS AND FUTURE WORK

In this chapter we presented a current overview of the DBpedia community project. With DBpedia, we also aim to provide a proof-of-concept and blueprint for the feasibility of large-scale knowledge extraction from crowd-sourced content repositories. There are a large number of further crowd-sourced content repositories and DBpedia already had an impact on their structured data publishing and interlinking. Two examples are Wiktionary with the Wiktionary extraction (Hellmann, Brekle, and Auer, 2012) meanwhile becoming part of DBpedia and *LinkedGeoData* (Stadler et al., 2012), which aims to implement similar data extraction, publishing and linking strategies for *OpenStreetMaps*.

4

INTERNATIONALIZATION OF DBPEDIA

Bratsas, Ioannidis, Kontokostas, Auer, Bizer, Hellmann, and Antoniou, (2011a) and Kontokostas, Bratsas, Auer, Hellmann, Antoniou, and Metakides, (2012)

The early versions of the *DBpedia Information Extraction Framework* (DIEF) used only the English Wikipedia as sole source. Since the beginning, the focus of DBpedia has been to build a fused, integrated dataset by integrating information from many different Wikipedia editions. The emphasis of this fused DBpedia was still on the English Wikipedia as it is the most abundant language edition. During the fusion process, however, language-specific information was lost or ignored. We establish best practices (complemented by software) that allow the DBpedia community[1] to easily generate, maintain and properly interlink language-specific DBpedia editions. We realized this best practice using the Greek Wikipedia as a basis and prototype and contributed this work back to the original DIEF. We envision the Greek DBpedia to serve as a hub for an emerging *Greek Linked Data* (GLD) Cloud (Bratsas et al., 2011b).

The Greek Wikipedia is, when compared to other Wikipedia language editions, still relatively small – 67th in article count[2] – with around 120,000 articles. Although the Greek Wikipedia is presently not as well organized – regarding infobox usage and other aspects – as the English one there is a strong support action by the Greek government[3] foreseeing Wikipedia's educational value to promote article authoring in schools, universities and by everyday users. This action is thus quickly enriching the GLD cloud. In addition, the Greek government, following the initiative of open access of all public data, initiated the geodata project,[4] which is publishing data from the public sector. The Greek DBpedia will not only become the core where all these datasets will be interlinked, but also provides guidelines on how they could be published, how non-Latin characters can be handled and how the Transparent Content Negotiations (TCN) rules (RFC 2295) (Holtman and Mutz, 1998) for de-referencing can be implemented.

In this chapter we present the results of the development and implementation of the internationalized DBpedia Information Extraction Framework (I18n-DIEF), which consists in particular of the following novelties:

1 The authors established the *DBpedia Internationalization Committee* to gather other interested community members aiming to create a network of internationalized DBpedia editions (http://dbpedia.org/internationalization).

2 Accessed on 08/08/2016: http://stats.wikimedia.org/EN/Sitemap.htm

3 http://advisory.ellak.gr/?p=12

4 http://geodata.gov.gr

1. Implementation of the DBpedia I18n Framework by plugging *I18n filters* into the original DBpedia framework (Section 4.1). This is now part of the official DBpedia Framework.
2. Use of DBpedia as a statistical diagnostic tool for Wikipedia correctness (Section 4.2.1 and Section 4.5).
3. Development of the *Template-Parameter Extractor* to facilitate a semi-automatic mapping add-on tool and provide the basis for the infobox-to-ontology mapping statistics (Section 4.2.2).
4. Justification of the need for language-specific namespaces (Section 4.3).
5. The linking of language specific DBpedia editions to existing link targets of the English DBpedia in the LOD Cloud (Section 4.3.1).
6. Definition of Transparent Content Negotiation rules for IRI dereferencing (Section 4.4.1).
7. Identification of IRI serialization problems and the proposal for an effective solution (Section 4.4.2).

SOLUTION OVERVIEW: I18N EXTENSION OF THE DBPEDIA INFORMATION EXTRACTION FRAMEWORK

In this section, we describe our I18n extensions of the DIEF, which improve I18n support of the software and introduce customizability features for language-specific Wikipedia editions. The Greek language is well suited for a I18n case study as a language with non-Latin characters and an average amount of articles. The described steps can be easily adjusted through configuration files for other language editions[5] and thus help to easily produce DBpedia datasets for each Wikipedia language edition. All described extension have been merged into the main DIEF trunk and are available as open source.[6]

The main rationale for facilitating better internationalization is to: (a) provide better customization options in extractors, that need to be customized for each language edition and (b) implement *I18n filters* where necessary, that can be plugged into the DIEF and *filter* the input and output of some components (cf. Figure 13).

In order to increase the amount and quality of the produced triples, the extractors had to be customized and tuned, where appropriate, for language-specific aspects. As illustrated in Figure 13 five extractors and four parser have been enhanced, namely:

5 http://wiki.dbpedia.org/Internationalization/Guide
6 https://sourceforge.net/projects/dbpedia/

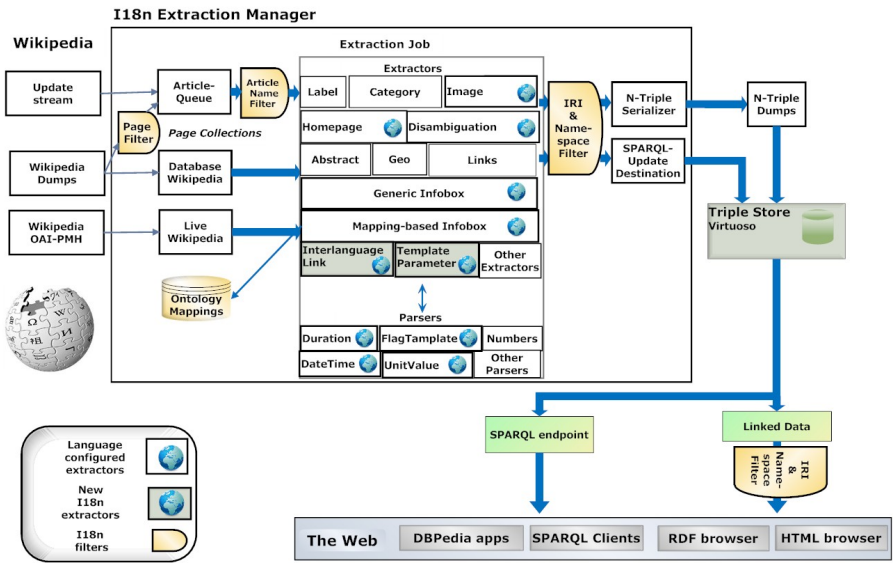


Figure 13: The internationalized DBpedia Information Extraction Framework including the I18n filters and the two new extractors.

DISAMBIGUATION EXTRACTOR extracts disambiguation pages and can be enabled by defining the language-specific token used by Wikipedia in the article title (e.g. "(disambiguation)" for the English Wikipedia[7]).

HOMEPAGE EXTRACTOR extracts links to the homepage of a resource by searching for the term "homepage" as a link title in the "External Links" section of an article. Other languages must provide translations for the "homepage" and "External links" terms (e.g. "Ιστοτοπος" and "Εξωτερικοι συνδεσμοι" respectively in Greek).

IMAGE EXTRACTOR generates links to *Wikimedia Commons* images. The extractor has to evaluate copyright templates in order to exclude "non-free" images for licensing issues. Our extractor extension allows to configure such language-specific templates.

GENERIC INFOBOX EXTRACTOR generates triples from infobox templates using properties from the `http://dbpedia.org/property/` namespace. Every Wikipedia language edition defines different infoboxes and properties and some of them do not provide any added value (e.g. width and height of an image, highlight colors, layout information, etc.), which thus can be configured to be excluded.

MAPPING-BASED INFOBOX EXTRACTOR extracts triples from infobox templates that are mapped to the DBpedia ontology and there-

7 http://en.wikipedia.org/wiki/Wikipedia:Disambiguation#Naming_the_disambiguation_page

fore uses properties from the `http://dbpedia.org/ontology/` namespace. Internationalization is achieved by defining template mappings (cf. Section 4.2).

DATE–TIME PARSER parses a date from a given string. We extended customizability of this parser in a way such that language-specific months, cardinality and era tokens can be provided.

DURATION PARSER parses durations from a given string. Other languages must provide translations for duration tokens (e.g. "λεπτα" for "minutes").

FLAG-TEMPLATE PARSER parses custom templates named after a country's code (e.g. {{GRE}} for Greece) that display the country's flag followed by the country link. Other languages must provide the corresponding country article for every flag template (e.g. "Ελλαδα" for {{GRE}}).

UNIT-VALUE PARSER parses units of measurement from a given string. Other languages must provide translations for different unit types (e.g. *m, meter, meters ,meter, μ, μετρα, μετρο* for the *meter* unit type).

In addition two new extractors were implemented for the I18n-DIEF:

INTERLANGUAGE-LINK EXTRACTOR creates RDF triples with the `dbpedia-owl:interlanguageLink` predicate between resources across different Wikipedia language editions. The Interlanguage-Link Extractor was an essential addition, since DBpedia's global naming namespace did previously not allow the representation of articles without an interlanguage link to an English article (cf. Section 4.3.1).

TEMPLATE-PARAMETER EXTRACTOR extracts the named parameters[8] that are declared by each Wikipedia template. The *Template-Parameter Extractor* was envisioned as an add-on tool that could facilitate a semi-automatic infobox-to-ontology mapping (cf. Section 4.2.2).

To further increase the number of extracted triples, we had to extract Greek articles without an English translation. By including all Greek articles, the triple count was increased by 41.7% (cf. Table 6). This, however, was not compatible with DBpedia's resource naming convention, as the untranslated articles could not be matched to the English namespace. This was the reason for defining language-specific resource namespaces (cf. Section 4.3). In this regard, four *filters* were created:

8 http://en.wikipedia.org/wiki/Help:Template#Handling_parameters

PAGE FILTER enables the addition of all articles in the article queue,

ARTICLE NAME FILTER enables the use of the original article name instead of the English translation,

NAMESPACE FILTER enables the extraction in a language-specific namespace and

IRI FILTER enables the extraction of resources in the IRI form.

The IRI form was adopted, replacing the commonly used URI, for reasons discussed in Section 4.4. As a result the resource names are differentiated because IRIs allow the use of UTF-8 characters, while URIs are limited to Latin characters and the percent-encoding of all other characters. For example, the IRI `http://el.dbpedia.org/resource/Χιος` would be represented in the URI form as `http://el.dbpedia.org/page/%CE%A7%CE%AF%CE%BF%CF%82`.

Virtuoso Universal Server (VUS)[9] was chosen as triple store for two reasons: firstly, VUS fully supports IRIs and UTF-8 according to (Auer et al., 2010) and secondly, VUS allows to use the same code-base of the English DBpedia for Linked Data publishing. This code-base was modified in order to handle IRIs (since the English DBpedia uses URIs) and to parametrically accept language-specific namespaces and graphs, apart from the default namespace and graph `http://dbpedia.org` (IRI and Namespace Filter). A thorough representation of the necessary steps is discussed in Section 4.4.1. Eventually, the software and Linked Data interfaces were completely compatible to the standard DBpedia as shown in Figure 14.

INFOBOX MAPPINGS AND PROPERTIES

One of the richest source of structured information in Wikipedia articles tapped by DBpedia are infobox templates.[10] For this purpose two kinds of extractors exist, namely: the *Generic Infobox Extractor* (cf. Section 3.1.3) and the *Mapping-Based Infobox Extractor* (cf. Section 3.1.4).

Greek Wikipedia case study

During the implementation process the following issues were tackled: (a) the infobox templates may not contain all the desired information, (b) some templates may be more abstract, thus cannot map to a specific class, (c) some templates are missing and (d) some templates are not used inside the articles. In order to statistically verify these issues, a slightly modified version of the *Generic Infobox Extractor* was developed. This Extractor generates a table with a row for every property,

9 http://virtuoso.openlinksw.com/
10 http://en.wikipedia.org/wiki/Help:Infobox

Figure 14: HTML representation using TCN rules.

consisting of the following columns: article name, template name, raw property name. Using this helper table we obtained statistics about parameter counts, template counts and property count per template. The results revealed, as was expected, many misspellings which led to errors and variations in property naming of the same entity across different templates.

For example, the entity 'record label' was referred to by the following parameter variations: δισκογραφικες εταιριες *(4)*, Δισκογραφικη *(99)*, δισκογραφικη *(46)*, Δισκογραφικη εταιρια *(1)*, δισκογραφικη εταιρια *(163)*, δισκογραφικη_εταιρια *(1)*, δισκογραφικηεταιρια *(1)* across six different templates due to orthography, case and gender variations.

We also detected the presence of multiple templates for the same topic, e.g. 50 articles using the template "Infobox Band" and 80 using the template "Μουσικο Συγκροτημα" which is the Greek equivalent of the English *band* infobox. This indicates the need for better coordination of the Greek Wikipedia community.

The results of our study were directly announced to the Greek Wikipedia community.[11] They were perceived as helpful and initiated a collective effort to correct the parameters and the templates. This feedback will eventually lead to better DBpedia results and also shows DBpedia's contribution to Wikipedia.

An effort was also initiated to rewrite Greek templates on the Greek Wikipedia taking not only the corresponding English template conventions into consideration but the DBpedia Ontology as well. In this context, the DBpedia Ontology is used as a common vocabulary, which could well develop as a standard ontology for data representa-

11 http://el.wikipedia.org/wiki/Bikipaideia:DBpedia/Metablhtes_15-12

tion across languages within Wikipedia. Although it is per se difficult to evaluate an ontology, the employment and reuse of the DBpedia Ontology by a second community can be evaluated as positively according to the criteria *Adaptability* and *Clarity* of (Vrandečić, 2010, p. 56) and therefore shows the usefulness and adequacy of the DBpedia Ontology.

Template-Parameter Extractor

Even though the extended mapping system is much more convenient than the previous one, the process still involves manual operations. In order to facilitate the mapping effort further, tools can be created in order to validate and (semi-)automatically link parameters and ontology properties. A step in this direction is the introduction of a new extractor that we implemented in the I18n-DIEF, the *Template-Parameter Extractor* (TPE). TPE extracts the parameters declared by each template by parsing them directly from the template pages and generates RDF triples using the `dbpedia-owl:template UsesParameter` predicate (cf. Listing 2). Template pages contain the template definitions and declare their parameters in wiki format using three consequent brackets (*{{{parameter name[|][default value]}}}*). Template parameters are actually the *allowed infobox properties* to be used by articles.

The infobox-to-ontology mapping statistics[12] is an existing application of the Template-Parameter Extractor (cf. Figure 15). The offline output of this extractor and the statistics from Section 4.2.1 are used to demonstrate the mapping coverage of the infobox templates and their parameters as well as how often parameters occur in a language-specific Wikipedia edition. Prospectively, this extractor can be exploited easily to develop semi-automatic tools for data curation, when combined with the DBpedia Live Extraction. All available infobox properties can for example be curated directly in a graphical mappings editor and periodical checks are possible, whether previously defined mappings are using a non-valid parameter (e.g. in case of a template is edited on Wikipedia and a new unmapped parameter was introduced).

LANGUAGE-SPECIFIC DESIGN OF DBPEDIA RESOURCE IDENTIFIERS

Currently, the fused DBpedia extracts non-English Wikipedia articles only when they provide an English interlanguage link and the created resources use the default DBpedia namespace (cf. Section 3.1.5). Although this approach minimizes the use of non-Latin characters in resource identifiers, it has a the following drawbacks:

12 http://mappings.dbpedia.org/index.php/Mapping_Statistics

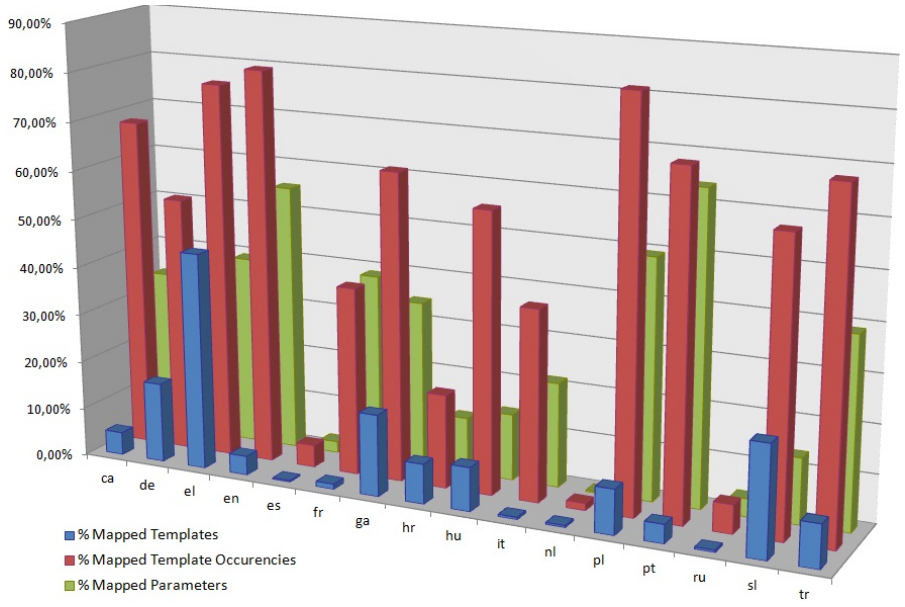


Figure 15: Infobox-to-ontology mapping statistics using the Template-Parameter Extractor as a core dataset

```
dbp-tpl:Infobox_book   dbp-owl:templateUsesParameter
        "name"@en, "title_orig"@en, "image"@en,
        "image_caption"@en, "author"@en,
        "illustrator"@en,  "cover_artist"@en .
```

Listing 2: Exampe dataset extracted with TPE for infobox book.

1. The merging is solely based on the link from the non-English resource to the English article. It has been shown that such links are more appropriate, if the interlanguage links go in both directions (Erdmann et al., 2008). Because we introduce `owl:sameAs` (a transitive property) to link between language editions, we especially conducted measurements to test the integrity of our design (cf. Section 4.3.1).

2. A large number of articles, without an English translation link, are discarded. For instance, the DIEF produces 30% less triples for the Greek DBpedia (cf. Table 6) than the I18n-DIEF.

3. The extracted non-English articles cannot provide information other than their abstract and label, as everything else either conflicts with an English definition or creates multiple definitions.

4. The English Wikipedia is treated as the authority, which may not be the case for language-specific articles. For instance, the article about the Eiffel Tower in the French Wikipedia[13] contains

13 http://fr.wikipedia.org/wiki/Tour_Eiffel [accessed on 2011/11/07].

more detailed information. Up to now, though, the English version of DBpedia was the only available option.

It is more appropriate that resources in non-English languages are published according to the Wikipedia's naming strategy, i.e. with the original article name, using a language-specific namespace (e.g. `http://el.dbpedia.org/` for Greek). As new languages are publishing their data, the English DBpedia might be transferred into `http://en.dbpedia.org/` and the default namespace could be used solely for the "Cross-language knowledge fusion" (Lehmann et al., 2009, p. 164).

Inter-DBpedia linking

Using the language-specific resource naming approach, an interlanguage link (ILL) can be utilized to connect resources across different DBpedia language editions and thus creates a multilingual semantic space. To accomplish this, a new extractor was developed for the I18n-DIEF, called *Interlanguage-Link (ILL) Extractor*. It extracts ILLs and generates RDF triples using the `dbpedia-owl:interlanguageLink` predicate. Using these links as a raw dataset, we examine whether they can be used to generate `owl:sameAs` links between resources extracted from different Wikipedia language editions. The ILL correspondence is not always reliable since by following ILLs across different languages, conflicts may appear (Bolikowski, 2009), as the following example illustrates:

`en:Tap (valve)` $\mapsto$ `it:Rubinetto` $\mapsto$ `es:Grifo` $\mapsto$ `en:Griffin`

We performed an analysis on the ILLs, which form a directed graph (V, E), where V is the set of Wikipedia pages as nodes and E is the set of ILLs between two pages which define the edges. Wikipedia mentions the following editor guideline: "An interlanguage link is mainly suitable for linking to the most closely corresponding page in another language".[14] Thus, each concept, represented as a set of Wikipedia pages, can be defined as a subgraph consisting of the corresponding pages in each language. When this subgraph contains *at most* one article from each language, the correspondence is *consistent*, otherwise we consider it a *conflict situation*. Using the simplified dataset provided by Bolikowski (Bolikowski, 2009),[15] the graph properties were re-calculated and presented in Table 5. From the results, we can estimate the extent of conflicts and whether the conflicts are reduced, if the ILL graph is restricted to two-way links only. The conflict analysis was performed using the English articles as starting point for the measurements since it is the largest dataset.

14 http://meta.wikimedia.org/wiki/Help:Interwiki_linking#Interlanguage_links

15 urlhttp://wikitools.icm.edu.pl/m/dumps/

Property	**Graph with all links**	**Graph restricted to two-way links**
Graph type	Directed	Undirected
Graph order (number of nodes)	878,333	825,764
Graph size (number of Links)	47,487,880	(2×) 23,001,554
Connected components (weak)	34,623	34,412
Conflicts (paths between two English articles)	380,902	(2×) 16,063
Different (English) *articles in conflicts*	5,400 (0.21%)	1,900 (0.07%)
Total English articles (as of August 2008)	2.5M	

Table 5: The ILL Graph Properties for all edges and for the subgraph of two-way edges. The calculations were performed with the open-source *R Project for Statistical Computing* (`http://www.r-project.org/`).

We observe that the relative error is very small: 0.21% of the total number of English articles are participating in conflicts, creating a total number of 380,902 conflicts. By restricting the graph to two-way links, 52,569 nodes and 1,484,772 edges are discarded. However, the discarded nodes (5.9%) are responsible for 65% of different English articles participating in conflicts, and the discarded links (3.1%) are responsible for 91.6% of the conflicts in English articles. The fact that the connected components are reduced from 34,623 to 34,412, i.e. 0.61%, is an indication that the graph structure does not change significantly, if the graph is restricted to two-way edges. Even though the relative error is small, by removing the one-way edges from ILLs, the conflicts are further reduced to 0.07% of the total number of English articles and the conflicts are reduced to 8.4%. The reason why we conducted the measurements are the strong semantical implications of `owl:sameAs`, as it produces equivalence classes in a multilingual network of language-specific DBpedia. This is why it is necessary to reduce any errors to a minimum. Our analysis indicates that the created conflicts are not significant if the `owl:sameAs` triples are considered only for two-way ILL edges.

In order to implement this analysis a new tool was created, that utilizes the ILL Extractor output from two languages, and generates `owl:sameAs` triples only for two-way edges. An example of a link extracted this way is:

dbr-el:Θεσσαλονικη **owl**:sameAs **dbp**:Thessaloniki

Additionally to the inter-DBpedia linking, another tool was developed that transitively links a non-English DBpedia to all the exter-

```
#For the English link:
dbr:Thessaloniki rdf:type geonames:734077.

#and the owl:sameAs link
dbr-el:Θεσσαλονικη owl:sameAs dbr:Thessaloniki

dbr-el:Θεσσαλονικη rdf:type geonames:734077
```

Listing 3: Example of the transitive linking to the LOD Cloud

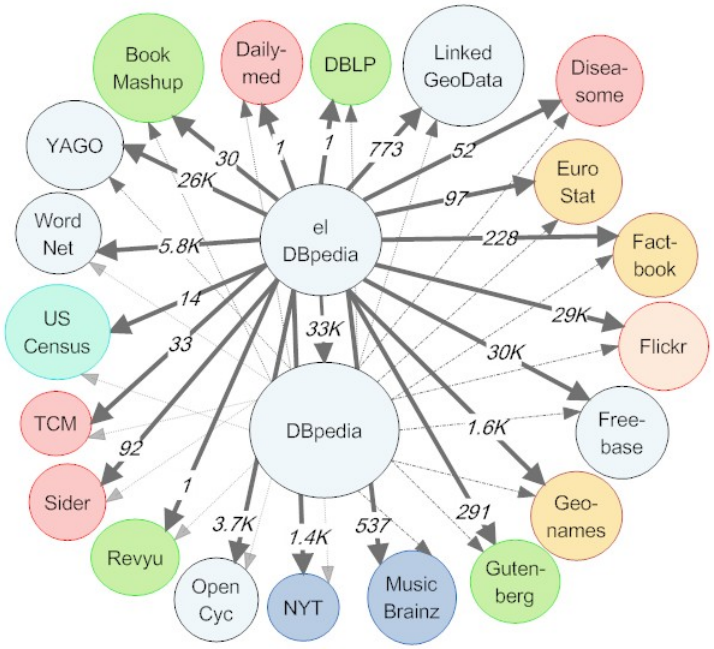


Figure 16: The Greek DBpedia in the I18n LOD Cloud.

nal LOD datasets that are linked to the English DBpedia (cf. Listing 3). Even though this could be accomplished using a SPARQL query (Prud'hommeaux, Harris, and Seaborne, 2013), this procedure would consume substantial server resources in querying and loading all the datasets. Our tool does not have this problem because the triples are created offline, directly in the N-Triples format.

In total, 33,148 `owl:sameAs` links to the English DBpedia were established (2339 links were only one-way and have been removed (6.59%)). As a result of our inter-DBpedia linking, a total of 101,976 additional `owl:sameAs` links were created, linking the Greek DBpedia with 20 external LOD datasets[16] (cf. Figure 16).

16 http://el.dbpedia.org/en/datasets

```
# SPARQL query with IRIs
PREFIX dbr: <http://el.dbpedia.org/resource/>
SELECT ?p ?o
WHERE {
  dbr:Αθηνα?p ?o. }

# SPARQL query with URIs
PREFIX dbr: <http://el.dbpedia.org/resource/>
SELECT ?p ?o
WHERE {
  dbr:%CE%91%CE%B8%CE%AE%CE%BD%CE%B1 ?p ?o. }
```

Listing 4: Simple SPARQL query about Athens (Greek:Αθηνα) using IRIs and URIs.

INTERNATIONAL RESOURCE IDENTIFIERS

Linked datasets are expected to provide both machine processable as well as user readable and interpretable content (e.g. an HTML representation) (Sauermann and Cyganiak, 2008). However, the requirements for readability (see e.g. lexvo's presentation of the term 'door'[17] and the translation links), and for manual SPARQL query construction (cf. Listing 4) in non-Latin languages such as Greek cannot be satisfied using URI's. Therefore, the only option presently available is to use International Resource Identifiers (IRIs) as defined in RFC 3987 (Duerst and Suignard, 2005).

IRI's are known to impose security issues, as certain characters from different languages appear identical to users (e.g. the Greek and Latin "A", "B", "K" characters correspond to different IRIs). In the Linked Data case, however, IRIs are mainly processed by machines and thus do not represent a security issue in most cases. Also, security issues are mitigated, since we do not advocate the use of non-Latin characters in the domain name part of IRIs and the internationalized datasets are published just by one authoritative source, i.e. the Greek DBpedia project under the domain `el.dbpedia.org`. Another issue concerning the IRI form is the lack of support by all triple serialization formats (Auer et al., 2010), which introduces difficulties in defining TCN rules (Holtman and Mutz, 1998) for IRI de-referencing purposes.

Transparent Content Negotiation Rules

The DBpedia Linked Data publication code is designed according to W3C guidelines (Sauermann and Cyganiak, 2008) and is responsible for handling all the URI requests. Depending on the client request, the client can get either a human readable XHTML/RDFa

17 http://www.lexvo.org/page/term/eng/door

representation of the resource, or a serialization in the desired format (e.g. RDF/XML, N-Triples, n3, etc.). This is achieved by defining proper Transparent Content Negotiation (TCN) rules (RFC 2295) for 303 content redirection. The default redirection for a DBpedia resource (`http://dbpedia.org/resource/*`) is the XHTML/RDFa representation and the client is redirected to `http://dbpedia.org/page/*`. For a resource serialization, the client may request the `http://dbpedia.org/data/*` URL for an RDF/XML serialization, or append a custom extension (i.e. n3) for a specific serialization format (e.g. `http://dbpedia.org/data/Thessaloniki.n3`). All the resource serialization requests are served by redirecting to an "on-the-fly" generated SPARQL DESCRIBE query Prud'hommeaux, Harris, and Seaborne, 2013, section 10.4, exported in the desired serialization format. However, in order to limit the page size, the XHTML/RDFa presentation is programmatically created. This is achieved by using multiple SPARQL SELECT queries, which limit the number of *objects* for every *predicate* to a maximum (i.e. 200).

In the I18n-DIEF, the DBpedia Linked Data publication sourcecode was modified in order to optionally serve resources from a different *domain* and *graph* (i.e. `http://el.dbpedia.org`). Additionally, changes were also applied in order to optionally serve IRIs instead of URIs. However, IRI content negotiation cannot be implemented as straightforward as URI content negotiation for two reasons: (1) the triple-store resource storage implementation and (2) the definition of the HTTP protocol (RFC 2616) (Fielding et al., 1999). Current triple-store implementations store the resource identifiers as strings, thus an IRI (containing non-Latin characters) and a corresponding URI referring to the same resource will not be equal. In the frame of the HTTP protocol, section 3.2 of the RFC states that the HTTP accepts only URI's, thus when a resource is requested, the server can only accept a URI request. Since the resources are stored as strings and the server can only accept URIs, three scenarios can be distinguished:

1. **Resources are stored in the URI form**: Both the HTTP request and the data are in the same format and everything will work as expected.

2. **Resources are stored in the IRI form**: The URI contained in the HTTP request for a certain resource has to be decoded into an IRI in order to be used subsequently. Since the XHTML/RDFa representation is programmatically created the request decoding can be easily handled. Other serializations, however, must be handled with care because the decoding must take place inside the TCN rules.

3. **Resources are stored in mixed form**: While this may be a rare case, the XHTML/RDFa representation could be accomplished

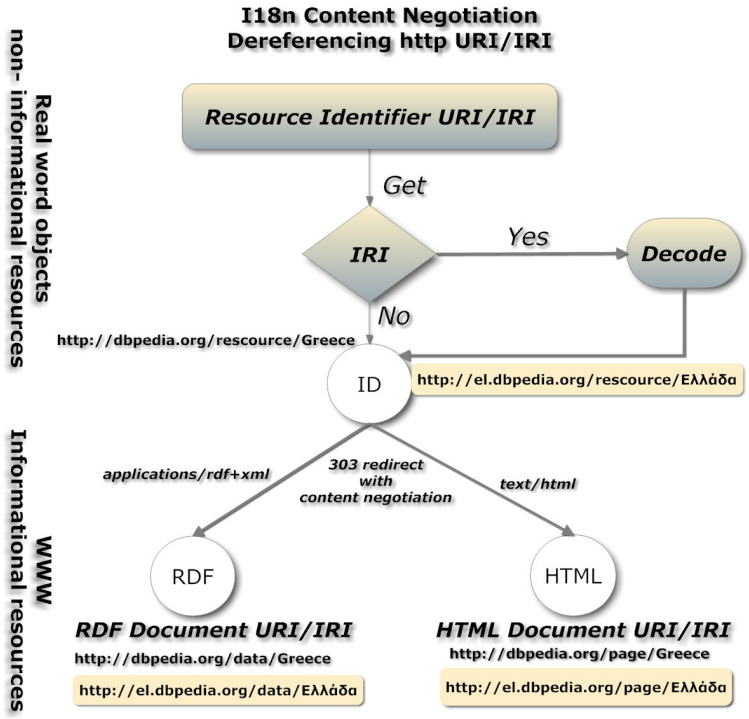


Figure 17: I18n Content Negotiation for URI and IRI dereferencing (modification of Sauermann and Cyganiak, 2008, Section 4.3).

using a SPARQL UNION of the normal and the decoded resource. For other serializations, the SPARQL DESCRIBE function cannot deliver all the triples and the only solution would be to programmatically re-implement the DESCRIBE function.

The I18n DBpedia Linked Data publication code is parametrizable, accepting both URIs and IRIs. Thus the same code is compatible both for an international DBpedia (using either IRIs or URIs) and the current fused DBpedia (using URIs). Furthermore, certain problems which appeared in cases where the UTF-8 characters were not encoded/decoded correctly were resolved. These problems were not addressed before because there was no need to do so as the previous version of DBpedia was restricted to Latin characters.

By encoding/decoding the HTTP request, we managed to workaround the imposed RFC limitations and provide de-referencable IRIs that could not be accomplished otherwise (cf. Figure 17).

IRI serialization

Among the most popular triple serialization formats only RDF/JSON (informal), RDFa (Adida et al., 2008) and Notation 3 (Berners-Lee and Connolly, 2008) are fully IRI compatible. N-Triples (Grant and

Becket, 2004) and N-Quads (Cyganiak, Harth, and Hogan, 2008) do not support IRIs at all, since they use 7-bit US-ASCII character encoding. [18] Turtle (Beckett, 2007) and RDF/XML (Manola and Miller, 2004) provide partial IRI support as their grammar definition is not fully mapped to the IRI grammar Duerst and Suignard, 2005, p. 7. In particular:

- In the frame of the *RDF/XML* representation, *subjects* and *objects* are defined as XML Attributes, therefore can be serialized as IRIs. *Predicates* on the other hand, must be declared as XML Elements. According to the XML specification XML Elements have many restrictions[19] on the allowed characters.

- *Turtle* can serialize IRIs but only when using absolute resources. The Turtle abbreviated form has certain limitations[20] that affect both IRIs and URIs.

Despite the mentioned limitations, triple stores are not so strict with the file formats and invalid serialization can be imported in some cases. However, depending on the format and the implementation, triple stores do not accept all IRI characters. For example, Virtuoso may accept IRIs in N/Triples and N/Quads files but without the '>' character; similar cases apply for other triple store implementations (cf. Auer et al., 2010, table 5). These characters were handled by encoding them using the URI %-encoding.

In order to solve IRI serialization issues, existing serialization standards should be adapted or new ones may be created. Apart from the serialization standards, triple stores could also provide a global solution to IRI issues by treating a URI and its corresponding IRI representation uniformly. Triple stores have "full knowledge or control" Duerst and Suignard, 2005, Section 5.1 of the resources, so it could be considered safe to use one of the two as an internal storage representation and transform input/output accordingly. This solution has, however, one limitation. Allowing to accept queries in both formats may require disambiguation in cases when a SPARQL query contains the URI escape character '%' in a resource. The advantage of this approach is to choose the resource representation according to the serialization format (e.g. URIs for N-Triples). The same solution applies for content negotiation rules as well, as no special cases have to be implemented to handle resource-encoding/decoding for the HTTP URI request.

18 This was true at the time of writing that paper, now utf-8 is allowed in both N-Triples and N-Quads.

19 http://www.w3.org/TR/REC-xml/#NT-Name

20 http://www.w3.org/TeamSubmission/turtle/#resource

STATISTICS AND EVALUATION

In this section we will look at the attained improvements due to the I18n revision of the DIEF. The results with regard to extracted triples for all available extractors (presented in Table 6) allow the comparison between the Greek and the English DBpedia editions. Extractions of the Greek Wikipedia with the I18n-DIEF refer to the same Wikipedia dump as the *Greek DBpedia v 3.5.1*. The final result of our efforts is presented in the column labeled **I18n-aa** (Greek DBpedia I18n-DIEF – **a**ll **a**rticles).

In the last row, the total number of extracted triples per page (*Triples/-PageID*) are listed. Despite the relatively small size of the Greek Wikipedia, the results indicate the equivalence of the Greek DBpedia to the English DBpedia with regard to average extracted triples per page (28.52 compared with 29.12). Furthermore, there is an increase by 62,6% in total triples,[21] when the I18n-DIEF is compared to the standard DBpedia DIEF for Greek (EL-3.5.1). The percentage increase is damped by the fact that many potential triples were not extracted from raw infobox properties (cf. Section 4.1), titles and links (DIEF restricted them only in articles pages) as they did not offer any added value. To compare our extraction on a per extractor basis with the English version, we calculated a **%-Diff** using the following formula:

$$\frac{\text{Triples}_{\text{EN}-3.5.1}}{\text{PageIDs}_{\text{EN}-3.5.1}} \; / \; \frac{\text{Triples}_{\text{I18n}-aa}}{\text{PageIDs}_{\text{I18n}-aa}} \; - \; 1$$

The resulting percentage signifies the increase in triples per page per extractor when comparing the English and the Greek version.

Although many factors influence the calculated %-diff (e.g. the implementation of the extractors), the value can be considered as an indicator for the differences in the structure of the English and the Greek Wikipedia. For instance: (a) –57% in ontology types means that infoboxes are used less frequently in the Greek Wikipedia, (b) –65% in Raw Infobox Property usage, means that infoboxes in the Greek Wikipedia, do not use as many properties as in the English Wikipedia. The issues (a) and (b) may be verified by manually examining translated articles directly.

The impact of the the I18n-DIEF in DBpedia 3.7 release is depicted in Figure 18. The percentage increase in triple count from the initial DBpedia 3.7 release to DBpedia 3.7 with the I18n extensions was 61.26%, only by enabling the *Article Name Filter* (cf. Section 4.1). As Figure 18 illustrates, this increase refers only to the 15 localized languages and ranges from 13.05% in Gabon (ga) to 84.27% in Russian (ru). This means that despite all the I18n extensions each language had already implemented, there exist many articles the do not have an English translation. With the previous framework, all this information would have been discarded.

21 1,677,656 compared to 2,679,214.

	EN-3.5.1	EL-3.5.1	I18n-ota	I18n-aa	%-Diff
Abstracts					
Extended	3,144,265	35,433	35,487	50,219	56.98%
Short	3,129,527	35,433	35,487	50,219	57.72%
Category					
Labels	565,108	-	6,062	9,577	66.57%
SKOS	2,247,931	-	22,442	34,872	52.47%
Articles	10,925,705	-	70,629	102,021	-8.22%
Geo Coordinates	1,462,892	14,424	14,360	18,760	26.04%
Images	4,203,605	-	58,205	67,520	57.87%
Links					
Disambiguation	768,924	-	4,302	7,233	-7.54%
External	5,081,932	-	61,652	77,981	50.82%
Homepages	394,897	-	1,221	1,354	-66.30%
Page	119,077,682	1,297,366	1,197,756	1,562,772	28.99%
to Article	21,997,875	130,614	108,924	221,724	-0.93%
Ontology Infobox					
Property usage	11,135,755	-	23,286	29,849	-73.65%
Specific properties	184,083	-	986	1,117	-40.36%
Types	5,495,590	-	17,822	24,087	-56.92%
Page IDs	9,042,227	-	43,375	91,997	0%
Person data	356,996	-	-	-	-
PND	1,297	-	-	-	-
Raw Infobox					
Property usage	38,119,014	117,066	105,077	135,740	-65.00%
Property definitions	77,124	3,782	3,576	3,718	373.83%
Redirects	4,082,533	-	104	22,549	-45.71%
Revision IDs	9,042,227	-	43,375	91,997	0%
Titles	7,332,625	43,538	36,308	73,908	-0.93%
Totals					
Triples	**257,869,814**	**1,677,656**	**1,890,436**	**2,679,214**	**2.12%**
Page IDs	**9,042,227**	**47,138**	**43,375**	**91,997**	**-**
Triples/Page ID	**28.52**	**35.59**	**43.58**	**29.12**	**-**

Table 6: Statistics of the extracted triples. Columns: (EN-3.5.1) English DBpedia v 3.5.1, (EL-3.5.1) Greek DBpedia v 3.5.1, (I18n-ota) Greek DBpedia I18n-DIEF - **o**nly **t**ranslated **a**rticles, (I18n-aa) Greek DBpedia I18n-DIEF - **a**ll **a**rticles, (%-Diff) Percentage difference per extractor (positive values lean towards the Greek DBpedia)

CONCLUSIONS

With the maturing of Semantic Web technologies proper support for internationalization is a crucial issue. This particularly involves the internationalization of resource identifiers, RDF serializations and corresponding tool support. The Greek DBpedia was the first step towards Linked Data internationalization and the first successful at-

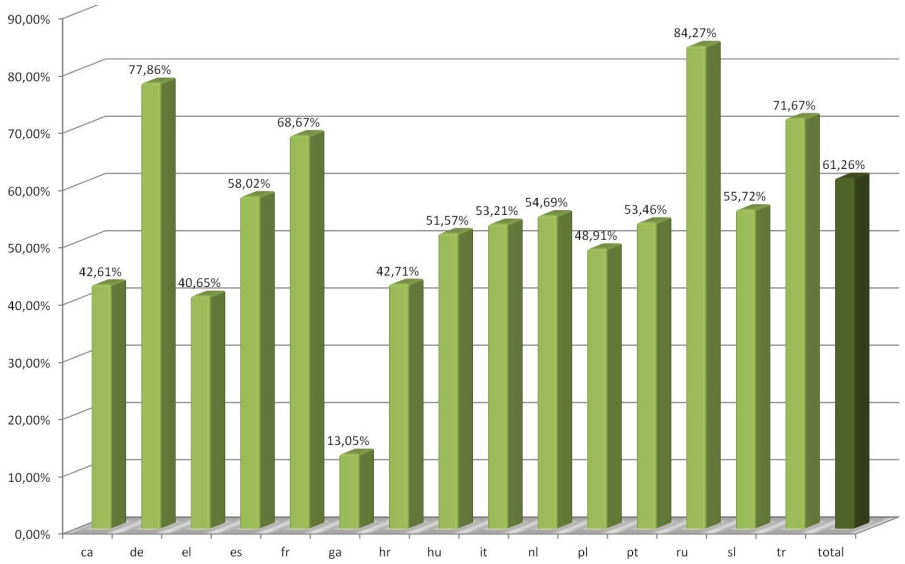


Figure 18: Percentage of increase in triples, just by enabling the *Article Name Filter* for the 15 localized languages in the DBpedia 3.7 release.

tempt to serve Linked Data with de-referencable IRIs that also serves as a guide for LOD publishing in non-Latin languages. Apart from the de-referencable IRI solution, this work provides the tools for a truly international DBpedia, as Greek is a comparatively complex language with non-Latin characters and non-standard punctuation. Since the Greek DBpedia provides qualitative information comparable to the English DBpedia, our I18n-DIEF can be easily transferred to other non-Latin Wikipedia editions and can (with slight language specific adoptions) be expected to give similar qualitative results.

As a result of our findings, the main DBpedia edition can also significantly contribute towards the IRI adoption. The switch of the English DBpedia edition as one major Linked Data hub to use IRIs will encourage other Linked Data providers to follow. Already 17.8% (i.e. 1,679,124 out of 9,485,630 in the 3.6 release) of all resources contain the % escape character and can therefore be simpler written as IRIs. An additional follow up of this work is the institutionalization of the Internationalization Committee Section 3.2.2

Another area of research is the more efficient utilization of the Wikipedia interlanguage links. The approach discussed in Section 4.3.1 was safe and straightforward. A further analysis of the *conflict situations* and how they could be resolved will be of great importance both for Wikipedia and the internationalization of the Semantic Web. The *conflict situations* analysis could also provide new data and make us re-examine the use of `owl:sameAs` – as a too strong semantic implication – with other vocabularies (i.e. SKOS). We could also utilize the *conflicts*, which are now discarded, by adding `rdfs:seeAlso` links.

Infobox Mappings will play a central role in the integration and evolution of international DBpedia editions. Developing better mapping tools is a crucial strategy to facilitate this process. The new "Template-Parameter Extractor" (Section 4.2.2) can be utilized in order to assist the ontology-to-infobox mappings definition and organization. In the next months we will obtain first indications on the results of the Wikipedia infobox restructuring and (re-)mapping (Section 4.2.1) in future DBpedia releases.

DBpedia's goal is to "make it easier for the amazing amount of information in Wikipedia to be used in new and interesting ways" and to "inspire new mechanisms for navigating, linking and improving the encyclopedia itself".[22] Our work provides new tools for improving Wikipedia, because DBpedia may serve as an important statistical diagnostic tool (cf. Section 4.2.1 and Section 4.5) for Wikipedia that helps to identify and resolve existing and emerging issues.

22 http://dbpedia.org

5

INCORPORATING WIKIMEDIA COMMONS IN DBPEDIA

Vaidya, Kontokostas, Knuth, Lehmann, and Hellmann, (2015)

Wikipedia is the largest and most popular open-source encyclopedia project in the world, serving 20 billion page views to 400 million unique visitors each month[1]. Wikipedia has over 200 language editions, from the English Wikipedia (4.6 million articles) to the Tumbuka Wikipedia (177 articles). Every article contains metadata such as its page title, its list of contributors, the categories it belongs to, and the other articles it links to. Articles may also contain structured data, such as the latitude and longitude of geographically situated articles. Since 2007, the DBpedia project has been extracting this metadata and structured data and making it publicly available as RDF (Lehmann et al., 2015).

Until recently, this extracted data had almost no information on the media files used to illustrate articles. While some media files are stored within a particular language edition of Wikipedia, over twenty-five million of them are located in a centralised repository known as Wikimedia Commons[2]. Wikimedia Commons acts as a media backend to all of Wikipedia; media files uploaded to it under an open-access license can be easily inserted into articles in any language. Metadata and structured data associated with the files are stored on Wikimedia Commons' MediaWiki instance, in a format similar to that used by Wikipedia. This will likely be superceded by Wikidata, the Wikimedia Foundation's new structured data store, but this project is still under discussion[3]. We make this large and well maintained media resource accessible for semantic tools by extending the DBpedia Extraction Framework to read data from Wikimedia Commons in addition to other Wikipedia language editions.

In this chapter we describe the dataset and the extraction process required to provide *DBpedia Commons* (DBc). We report on the extensions on the DBpedia Information Extraction Framework (DIEF) to support Media pages, multiple languages on the same page, and proper Wikimedia Commons media URL construction. In addition we describe the ontological changes we made in the DBpedia ontology for annotating media files and the additional external vocabularies we chose for the media representation. To our knowledge, this is the first complete RDFization of Wikimedia Commons and the largest media metadata RDF database in the LOD cloud.

1 http://reportcard.wmflabs.org/

2 http://commons.wikimedia.org/

3 See https://commons.wikimedia.org/wiki/Commons:Wikidata and https://www.mediawiki.org/wiki/Multimedia/Structured_Data.

WIKIMEDIA COMMONS

Wikimedia Commons follows many of the same conventions as Wikipedia itself: regular pages can contain textual content and embedded media files, pages may be placed in more than one category, and namespaces allow project and policy pages to be separated from content pages. Two main differences distinguish Wikimedia Commons from Wikipedia:

1. Every Wikipedia edition is written entirely in a single language. Wikimedia Commons is designed to be used by users of every language: where possible, page content is written in multiple languages so that it can be understood by these users. A series of templates allows the correct language to be displayed based on the users' browser and MediaWiki settings.

2. Most Wikipedia content is in its page content, i.e. its articles. Most Wikimedia Commons content is associated with individual files in the `File` namespace: thus, rather than describing a subject, as Wikipedia articles do, most Wikimedia Commons content describes a media file.

Our strategy for extracting data from Wikimedia Commons content therefore focused on extracting as much information as possible for each page from the `File` namespace. Since the DBpedia Extraction Framework can already extract content from MediaWiki archival dumps, we decided to modify it to support extracting content from archival dumps of Wikimedia Commons[4]. Note that this means the extraction framework never examines the media files directly; instead, it uses MediaWiki's dump format to infer statements about them.

WIKIMEDIA COMMONS EXTRACTION

We have identified three kinds of data that we are interested in: (1) File Metadata, (2) Page Metadata, and (3) Content Metadata. File metadata describes the file that has been uploaded to Wikimedia Commons, such as its encoding format and file size; these are stored in the backend database used by the MediaWiki software that runs the Wikipedia websites. Page metadata is stored for each MediaWiki page, including those that describe files. This includes the page title, the list of contributors, and a history of changes. Finally, the content metadata is stored on the MediaWiki page itself: this includes a list of outgoing external and internal links, the list of categories the page belongs to as well as standard templates that allowed descriptions, sources, authority information and latitude and longitude for the subject of the page to be stored. After investigating the available

4 Such dumps are created monthly at http://dumps.wikimedia.org/commonswiki/.

file metadata[5], we decided to focus on Page and Content Metadata. Except for image size, width and height, we could get the rest of the information from the Page and Content metadata.

The DBpedia Information Extraction Framework (DIEF) has support for reading MediaWiki XML exports. DIEF was modified to read monthly backups of Wikimedia Commons. Many of the extractors used to extract page metadata functioned flawlessly on the Wikimedia Commons dump, extracting titles, categories, authors and other page and content metadata into RDF with only minor changes (Kontokostas et al., 2012). Four new File Extractors targeting Wikimedia Commons specific information were developed (Section 5.2.1). The DBpedia mapping-based extractor was adapted to work on Wikimedia Commons media and creator pages (Section 5.2.2). We used this extractor to obtain licensing information through the mapping-based extraction.

IRI SCHEME By using the `http://commons.dbpedia.org` domain and following the existing naming strategy of DBpedia, the DBc resources are published under the `http://commons.dbpedia.org/resource/` namespace. For example, `http://commons.dbpedia.org/resource/File:DBpediaLogo.svg`.

Media Extractors

Four new DBpedia extractors were developed to capture the additional structures of Wikimedia Commons compared to the various Wikipedia language editions.

FileTypeExtractor

The FileTypeExtractor contains a preconfigured media index in order to guess the media MIME type based on the media extension. Depending on the file extension, the resource is assigned a direct `rdf:type` and the transitive closure of the direct type (cf. Figure 19 and Section 5.3). The direct type is also linked with `dct:type`. `dct:format` captures the MIME type according to RFC 6838[6]. The file extension is directly queryable with `dbo:fileExtension`. In addition, we provide `dbo:fileURL` for access to the final media URL and `dbo:thumbnail` and `foaf:depiction` for still images. This extractor also provides links to the image itself by using the special page `Special:FilePath`, which provides redirects to the image file. A sample output of this extractor is:

```
1 dbr-com:DBpediaLogo.svg a dbo:StillImage, dbo:Image, foaf:Image,
```

5 A detailed description is available at `https://www.mediawiki.org/wiki/Manual:Image_table`.

6 RFC6838 is available at `http://tools.ietf.org/html/rfc6838`.

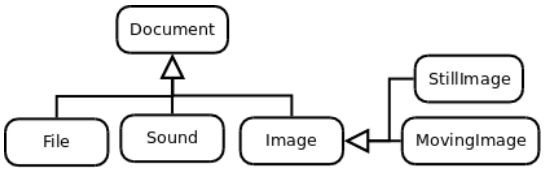


Figure 19: Hierarchy of main Document classes for DBpedia Commons

```
  schema:CreativeWork, dbo:File, dbo:Document,
  foaf:Document, dbo:Work ;
dct:type dbo:StillImage
dct:format "image/svg+xml";
dbo:fileExtension "svg" ;
dbo:fileURL commons-path:DBpediaLogo.svg;
dbo:thumbnail commons-path:DBpediaLogo.svg?width=300;
foaf:depiction commons-path:DBpediaLogo.svg.
```

GalleryExtractor

Wikimedia Commons and Wikipedia both support galleries, which makes it easy to display a series of related images in a compact format[7]. On Wikimedia Commons, this may be used to display a representative set of images about a single topic, such as the page for Colorado[8]. The GalleryExtractor identifies galleries embedded in pages, extracts the list of individual media items, and links them to the page resource with `dbo:galleryItem`.

```
dbr-com:Colorado dbo:galleryItem
  dbr-com:2006_CO_Proof.png, dbr-com:Anasazi_Heritage_Center.jpg,
  dbr-com:Bearlakeinspring2.jpg, dbr-com:Beol_court25.jpg .
```

ImageAnnotationExtraction

Wikimedia Commons may contain additional annotations for parts of a still image. The annotations mark a rectangular region in the picture and provide a description text in MediaWiki syntax, which may in turn contain hyperlinks to other resources. We managed to extract this information using the W3C *Media Fragments* (Troncy et al., 2012) vocabulary. We identify the annotated box by a separate IRI that is linked to the original resource through the `dbo:hasAnnotation`. As seen in the example below, the new IRI is based on the original resource by suffixing the image dimensions and the coordinates of the box. The image dimensions are required in case the original image needs to be scaled.

```
@prefix ann: <http://commons.wikimedia.org/wiki/Special:FilePath/
    Yes_concert.jpg?width=1514&height=1024#xywh=pixel:539,380,110,108>.
dbr-com:Yes_concert.jpg dbo:hasAnnotation ann: .
```

7 http://en.wikipedia.org/wiki/Help:Gallery_tag
8 http://commons.wikimedia.org/wiki/Colorado

```
ann: "Jon Anderson"@en .
```

CommonsKMLExtractor

Wikimedia Commons contains, in addition to media files, KML files as well. KML[9] is an XML format used for describing map overlays. The CommonsKMLExtractor extracts the KML data from Wikimedia Commons and stores them as an `rdfs:XMLLiteral` value of the `dbo:hasKMLData` property.

```
dbr-com:Yellowstone_1871b.jpg dbo:hasKMLData """
  <?xml version=1.0 encoding=UTF-8?>
  <kml xmlns="http://earth.google.com/kml/2.2">
   <GroundOverlay> <!-- KML data --> </GroundOverlay></kml>"""^^rdfs:
       XMLLiteral .
```

Infobox to Ontology Mappings

DBpedia has a sophisticated system for extracting infoboxes from Wikipedia articles. An 'infobox' is a special template that stores semi-structured data about the subject of an article. For example, an 'Infobox person' may record the birth date and location of the person, while an 'Infobox book' might record the ISBN and OCLC number of the book. A similar set of templates provides information on Wikimedia Commons where some allow for structured data to be inserted into a document. For example, the 'Location' template, which stores the location that a media file was created. In some cases, this structured data might be very rich, such as the 'NARA-image-full' template used to annotate images uploaded by the National Archives and Records Administration (NARA) of the United States, which contains Archival Research Catalog identifiers for each image.

A new DBpedia mapping namespace for Wikimedia Commons was created[10] and DIEF was refactored to extract mappings from media file and creator pages.

License Extraction

Using the mapping-based extraction we managed to extract license information from media files. Licenses are encoded in Wikimedia Commons as special templates, e.g. `{{cc-by-sa}}`. An initial approach was to map each license template to `owl:Thing` and provide a constant mapping to the license IRI to be able to reuse the existing mapping infrastructure. However, it is a common practice in Wikimedia Commons to nest and embed multiple licenses together and nested or wrapped templates are not supported in DIEF. To overcome this we

9 https://developers.google.com/kml/documentation/
10 http://mappings.dbpedia.org/index.php/Mapping_commons

Title	Triples	Description
Labels	29,203,989	Labels for resources
Provenance	272,079,712	Provenance information (pageIDs, revisionIDs)
SKOS	94,701,942	SKOS hierarchy based on the category hierarchy
Geo data	18,400,376	Geo coordinates for the media files
File Information	414,118,159	File metadata
Annotations	721,609	Image annotations
Galleries	2,750,063	Image galleries...
Types	111,718,049	Resource types
KML	151	KML data
Mappings	95,733,427	Mapped infobox data
Infobox	87,846,935	Unmapped Infobox data
Interlanguage links	4,032,943	Links to other DBpedia editions
Internal links	116,807,248	Internal links to other Wikimedia Commons pages
External links	17,319,980	Links to external resources
Metrics	58,407,978	Article metadata (in/out degree, page size)
Templates	77,220,130	Template metadata and usage

Table 7: Description of the DBc datasets

introduced a pre-processing extraction step to unwrap license templates and make all license mappings identifiable to the mapping extractor.

```
dbr-com:DBpediaLogo.svg dbo:license <http://creativecommons.org/
    publicdomain/mark/1.0/>
```

DATASET

A general overview of the datasets provided by DBc is provided in Table 7, where for each row provides a summary of one or more similar datasets. A total of 1.4 billion RDF triples are provided describing almost 30 million unique IRIs.[11] A diagram for the new classes we introduced for Wikimedia Commons media files is depicted in Figure 19. `dbo:Document` has the subclasses `dbo:File`, `dbo:Sound`, and `dbo:Image`. A `dbo:Image` can be a `dbo:StillImage` (e.g. picture) or a `dbo:MovingImage` (e.g. video). A complete hierarchy of the DBpedia ontology with the properties attached to each class can be found in the DBpedia mappings wiki.[12] According to Table 8, DBc mostly consists of still images with `jpeg` as the most popular format (cf. Ta-

11 All the datasets and the following statistics are based on Wikimedia Commons dump from January 2015

12 http://mappings.dbpedia.org/server/ontology/classes/#Document

Count	Class
25,061,835	dbo:StillImage
611,288	dbo:Artwork
90,011	dbo:Agent
49,821	dbo:MovingImage
19,126	dbo:Person

Table 8: Top classes

Count	Property
73,438,813	dct:subject
43,209,414	dbo:license
29,201,812	dce:language
24,496,724	dbo:fileURL
24,496,706	dbo:fileExtension

Table 9: Top properties

Count	MIME type
20,880,240	image/jpeg
1,457,652	image/png
878,073	image/svg+xml
455,947	image/tiff
354,873	application/octet-stream

Table 10: Top MIME types

Count	License
7,433,235	CC-by-sa v3.0
4,096,951	CC-pd v1.0
3,704,043	GNY-fdl v1.2
3,681,840	GNU-fdl
2,116,411	CC-by-sa v2.0

Table 11: Top licenses

ble 10). Table 9 provides the most frequent properties in DBc while Table 11 the most common media licenses. One of the largest datasets are the *mappings* (414M triples) which is based on the infobox to ontology mappings in the DBpedia mapping wiki. The authors, with contributions from the DBpedia community, invested significant effort to achieve this and managed to obtain a mapping coverage of 90% for all infobox template occurrences and 78% for all property occurrences.[13]

Access and Sustainability

DBpedia Commons is part of the official DBpedia knowledge infrastructure and is published through the regular releases of DBpedia, along with the rest of the DBpedia language editions. The first DBpedia release that included this dataset is *DBpedia 2014*[14]. DBpedia is a pioneer in adopting and creating best practices for Linked Data and RDF publishing. Thus, being incorporated into the DBpedia publishing workflow guarantees: a) long-term availability through the DBpedia Association and the Leipzig Computer Center long term hosting platform and b) agility in following best-practices as part of the DBpedia Information Extraction Framework.

Besides the stable dump availability we created `http://commons.dbpedia.org` for the provision of a Linked Data interface (Lukovnikov et al., 2014), a SPARQL Endpoint[15] and more frequent dataset up-

13 Retrieved 25/4/15 `http://mappings.dbpedia.org/server/statistics/commons/`
14 `http://downloads.dbpedia.org/2014/commons`
15 `http://commons.dbpedia.org/sparql`

dates[16]. The dataset is registered in DataHub[17]and provides machine readable metadata as void[18] and DataID[19] (Brümmer et al., 2014). Since the project is now part of the official DBpedia Information Extraction Framework, our dataset reuses the existing user and developer support infrastructure, e.g. the general discussion and developer list as well as an issue tracker[20] for submitting bugs.

USE CASES

In the following, we provide several existing or possible use cases of the DBc dataset.

DIGITAL LIBRARIES Galleries, Libraries, Archives and Museums (GLAMs) have always been interested in digital media as well as their metadata. Several GLAMs are already contributing media and media metadata in Wikimedia Commons.[21] So far they were mostly able to contribute data but not able to efficiently query for media other than their own. DBc is able to change that by allowing sophisticated SPARQL queries on the whole Wikimedia Commons metadata.

IMAGE RECOGNITION ALGORITHMS Using the Image annotation dataset we extract image area annotations. This dataset can be used as a training dataset for feature extraction from images. By using the links or the annotation description one could potentially link the annotated areas with external IRIs or resources from the LOD cloud.

META-DATA BASED MEDIA SEARCH In particular, the German National Library, contacted us regarding DBc. They were using Wikimedia Commons images in their Linked Data interface and were interested in displaying the license information directly there through our license dataset. The integration though is not yet deployed.

CONCLUSIONS AND FUTURE WORK

In this chapter we introduced DBpedia Commons, the first – to our knowledge - large-scale knowledge extraction from Wikimedia Commons. We present the adaptations and additions in the DBpedia Information Extraction Framework to facilitate the correct extraction of media files and license information. The dataset contains 1.4 billion RDF triples that provide file metadata, provenance, descriptions, and license information.

16 http://commons.dbpedia.org/data_access
17 http://datahub.io/dataset/dbpedia-commons
18 http://commons.dbpedia.org/void.ttl
19 http://dbpedia.s16a.org/commons.dbpedia.org/20150110/dataid.ttl
20 https://github.com/dbpedia/extraction-framework/issues
21 http://commons.wikimedia.org/wiki/Commons:GLAM

6

INCORPORATING WIKIDATA IN DBPEDIA

Ismayilov, Kontokostas, Auer, Lehmann, and Hellmann, (2016)

In the past decade, several large and open knowledge bases were created. A popular example, DBpedia (Lehmann et al., 2015), extracts information from more than one hundred Wikipedia language editions and Wikimedia Commons (Vaidya et al., 2015) resulting in several billion facts. A more recent effort, Wikidata (Vrandečić and Krötzsch, 2014), is an open knowledge base for building up structured knowledge for re-use across Wikimedia projects.

At the time of writing, both databases grow independently. The Wikidata community is manually curating and growing the Wikidata knowledge base. The data DBpedia extracts from different Wikipedia language editions, and in particular the infoboxes, are constantly growing as well. Although this creates an incorrect perception of rivalry between DBpedia and Wikidata, it is on everyone's interest to have a common source of truth for encyclopedic knowledge. Currently, it is not always clear if the Wikidata or the Wikipedia community provide more up-to-date information. In addition to the independent growth of DBpedia and Wikidata, there is a number of structural complementarities as well as overlaps with regard to identifiers, structure, schema, curation, publication coverage and data freshnenss that are analysed throughout this manuscript.

We argue that aligning both knowledge bases in a loosely coupled way would produce an improved resource and render a number of benefits for the end users. Wikidata would have an alternate DBpedia-based view of its data and an additional data distribution channel. End users would have more options for choosing the dataset that fits better in their needs and use cases. Additionally, it would create an indirect connection between the Wikidata and Wikipedia communities that could enable a big range of use cases.

The remainder of this chapter is organized as follows. Section 6.1 provides an overview of Wikidata and DBpedia as well as a comparison between the two datasets. Following, Section 6.2 provides a rational for the design decision that shaped DBw, while Section 6.3 details the technical details for the conversion process. A description of the dataset is provided in Section 6.4 followed with statistics in Section 6.5. Access and sustainability options are detailed in Section 6.6 and Section 6.7 discusses possible use cases of DBw. Finally, we conclude in Section 6.8.

COMPARISON AND COMPLEMENTARITY

The wikidata knowledge base is thoroughly described in Section 7.1.1. Both DBpedia and Wikidata knowledge bases overlap as well as complement each other as described in the high-level overview below.

IDENTIFIERS DBpedia uses human-readable Wikipedia article identifiers to create IRIs for concepts in each Wikipedia language edition. Wikidata on the other hand uses language-independent numeric identifiers.

STRUCTURE DBpedia starts with RDF as a base data model while Wikidata developed its own data model, which provides better means for capturing provenance information. Using the Wikidata data model as a base, different RDF serializations are possible.

SCHEMA Both schemas of DBpedia and Wikidata are community curated and multilingual. The DBpedia schema is an ontology based in OWL that organizes the extracted data and integrates the different DBpedia language editions. The Wikidata schema avoids direct use of RDFS or OWL terms and redefines many of them, e.g. `wkd:P31` defines a local property similar to `rdf:type`. There are attempts to connect Wikidata properties to RDFS/OWL and provide alternative exports of Wikidata data.

CURATION All DBpedia data is automatically extracted from Wikipedia and is a read-only dataset. Wikipedia editors are, as a side-effect, the actual curators of the DBpedia knowledge base but due to the semi-structured nature of Wikipedia, not all content can be extracted and errors may occur. Wikidata on the other hand has its own direct data curation interface called WikiBase,[1] which is based on the MediaWiki framework.

PUBLICATION Both DBpedia and Wikidata publish datasets in a number of Linked Data ways, including datasets dumps, dereferencable URIs and SPARQL endpoints.

COVERAGE DBpedia provides identifiers for all structural components in a Wikipedia language edition. This includes articles, categories, redirects and templates. Wikidata creates common identifiers for concepts that exist in more than one language. For example, not all articles, categories, templates and redirects from a Wikipedia language edition have a Wikidata identifier. On the other hand, Wikidata has more flexible notability criteria and can describe concepts beyond Wikipedia. There has not yet been a thorough qualitative and quantitative comparison in terms of

1 http://wikiba.se/

content but the following two studies provide a good comparison overview (Färber et al., 2016; Paulheim, 2016).

DATA FRESHNESS DBpedia is a static, read-only dataset that is updated periodically. An exception is DBpedia Live (available for English, French and German). On the other hand, Wikidata has a direct editing interface where people can create, update or fix facts instantly. However, there has not yet been a study that compares whether facts entered in Wikidata are more up to date than data entered in Wikipedia (and thus, transitively in DBpedia live).

CHALLENGES AND DESIGN DECISIONS

In this section we describe the design decisions we took to shape the DBpediaWikidata (DBw) dataset while maximising compatibility, (re)usability and coherence with regard to the existing DBpedia datasets.

NEW IRI MINTING The most important design decision we had to take was whether to re-use the existing Wikidata IRIs or minting new IRIs in the DBpedia namespace. The decision dates back to 2013, when this project originally started and after lengthy discussions we concluded that minting new URIs was the only viable option.[2] The main reason was the impedance mismatch between Wikidata data and DBpedia as both projects have minor deviations in conventions. Thus, creating new IRIs allows DBpedia to make local assertions on Wikidata resources without raising too many concerns.

RE-PUBLISHING MINTED IRIS AS LINKED DATA Since 2007, there has been many tools created by the DBpedia community to explore and exploit DBpedia data through the DBpedia ontology. Although there does not exists any thorough survey, some of these tools are collected on the DBpedia website and we refer the readers to publications related to DBpedia. [3] The decision to publish DBw, enables those tools that are designed to consume the DBpedia ontology to be able to consume, hopefully out of the box, the Wikidata data as well. One other main use case for publishing the DBw dataset is the creation of a new fused version of the Wikimedia ecosystem that integrates data from all DBpedia language editions, DBpedia Commons and Wikidata. Normalizing datasets to a common ontology is the first step towards data integration and fusion. Most companies (e.g. Google, Yahoo, Bing, Samsung) keep these datasets hidden; however,

2 http://www.mail-archive.com/dbpedia-discussion@lists.sourceforge.net/msg05494.html

3 https://scholar.google.gr/scholar?hl=en&q=DBpedia

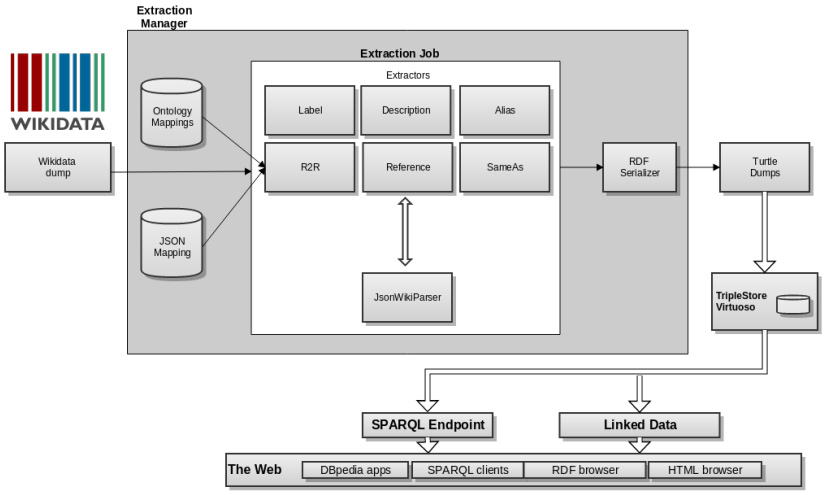


Figure 20: DBw extraction architecture

our approach is to keep all the DBpedia data open to the community for reuse and feedback.

ONTOLOGY DESIGN, REIFICATION AND QUERYING The DBpedia ontology is a crowdsourced ontology developed and maintained since 2006. The DBpedia ontology has reached a stable state where mostly additions and specializations are added in the ontology. At the time of writing, the DBpedia ontology defines 375 datatypes and units.[4] The Wikidata schema on the other hand is quite new and evolving and thus, not as stable. Simple datatype support in Wikidata started from the beginning of the project but units were only introduced at the end of 2015. In addition, Wikidata did not start with RDF as a primary data representation mechanism. There were different RDF serializations of Wikidata data and in particular different reification techniques. For example the RDF we get from content negotiation[5] is still different from the RDF dumps[6] and the announced reification design (Erxleben et al., 2014). For these reasons we chose to use the DBpedia ontology and simple RDF reification. Performance-wise neither reification techniques brings any great advantage (Hernandez, Hogan, and Krötzsch, 2015) and switching to the Wikidata reification scheme would require to duplicate all DBpedia properties.

4 http://mappings.dbpedia.org/index.php?title=Special:AllPages&namespace=206

5 https://www.wikidata.org/entity/Q42.nt

6 http://tools.wmflabs.org/wikidata-exports/rdf/

CONVERSION PROCESS

The DBpedia Information Extraction Framework was greatly refactored to accommodate the extraction of data in Wikidata. The major difference between Wikidata and the other Wikimedia projects DBpedia extracts is that Wikidata uses JSON instead of WikiText to store items.

In addition to some DBpedia provenance extractors that can be used in any MediaWiki export dump, we defined 10 additional Wikidata extractors to export as much knowledge as possible out of Wikidata. These extractors can get labels, aliases, descriptions, different types of sitelinks, references, statements and qualifiers.

For statements we define a RawWikidataExtractor that extracts all available information but uses our reification scheme (cf. Section 6.4) and the Wikidata properties and the R2RWikidataExtractor that uses a mapping-based approach to map Wikidata statements to the DBpedia ontology. Figure 20 depicts the current DBw extraction architecture.

Wikidata Property Mappings

In the same way the DBpedia mappings wiki defines infobox to ontology mappings, in the context of this work we define Wikidata property to DBpedia ontology mappings. Wikidata property mappings can be defined both as *Schema Mappings* and as *Value Transformation Mappings*. Related approaches have been designed for the migration of Freebase to Wikidata (Tanon et al., 2016).

Schema Mappings

The DBpedia mappings wiki[7] is a community effort to map Wikipedia infoboxes to the DBpedia ontology and at the same time crowd-source the DBpedia ontology. Mappings between DBpedia properties and Wikidata properties are expressed as `owl:equivalentProperty` links in the property definition pages, e.g. `dbo:birthPlace` is equivalent to `wkdt:P569`.[8] Although Wikidata does not define classes in terms of `RDFS` or `OWL` we use `OWL` punning[9] to define `owl:equivalentClass` links between the DBpedia classes and the related Wikidata items, e.g. `dbo:Person` is equivalent to `wkdt:Q5`.[10]

7 http://mappings.dbpedia.org
8 http://mappings.dbpedia.org/index.php/OntologyProperty:BirthDate
9 https://www.w3.org/TR/owl2-new-features/#F12:_Punning
10 http://mappings.dbpedia.org/index.php/OntologyClass:Person

Value Transformations

The value transformation takes the form of a JSON structure that binds a Wikidata property to one or more value transformation strings. A complete list of the existing value transformation mappings can be found in the DIEF. [11] The value transformation strings that may contain special placeholders in the form of a '$' sign represent transformation functions. If no '$' placeholder is found, the mapping is considered constant. e.g. `"P625": {"rdf:type": "geo:SpatialThing"}`. In addition to constant mappings, one can define the following functions:

$1 replaces the placeholder with the raw Wikidata value. e.g. `"P1566": {"owl:sameAs": "http://sws.geonames.org/$1/"}`.

$2 replaces the placeholder with an escaped value to form a valid MediaWiki title, used when the value is a Wikipedia title and needs proper whitespace escaping. e.g. `"P154": {"logo": "http://commons.wikimedia.org/wiki/Special:FilePath/$2"}`.

$GETDBPEDIACLASS Using the schema class mappings, tries to map the current value to a DBpedia class. This function is used to extract `rdf:type` and `rdfs:subClassOf` statement from the respective Wikidata properties. e.g `"P31": {"rdf:type": "$getDBpediaClass"} "P279":{"rdfs:subClassOf": "$getDBpediaClass"}`

$GETLATITUDE, $GETLONGITUDE, $GETGEORSS Geo-related functions to extract coordinates from values. The following is a complete geo mapping that the extracts geo coordinates similar to the DBpedia coordinates dataset.[12]

For every occurrence of the property P625, four triples — one for every mapping — are generated:

```
"P625":[{"rdf:type":"geoSpatialThing"},
 {"geo:lat": "$getLatitude" },
 {"geo:long": "$getLongitude"},
 {"georss:point": "$getGeoRss"}]
```

Listing 5: Geographical DBw mappings

```
dw:Q64 rdf:type geo:SpatialThing ;
 geo:lat "52.51667"^^xsd:float ;
 geo:long "13.38333"^^xsd:float ;
 geo:point "52.51667 13.38333" .
```

Listing 6: Resulting RDF from applied mappings for Wikidata item Q64

11 https://github.com/dbpedia/extraction-framework/blob/master/dump/config.json

12 http://wiki.dbpedia.org/Downloads

MAPPINGS APPLICATION The *R2RWikidataExtractor* merges the schema and value transformation property mappings. For every statement or qualifier it encounters, if mappings for the current Wikidata property exist, it tries to apply them and emit the mapped triples. Statements or qualifiers without mappings are discarded by the *R2RWikidataExtractor* but captured by the *RawWikidataExtractor* (cf. Section 6.4).

Additions and Post Processing Steps

Besides the basic extraction phase, additional processing steps are added in the workflow.

TYPE INFERENCING In a similar way DBpedia calculates transitive types for every resource, the DBpedia Information Extraction Framework was extended to generate these triples directly at extraction time. As soon as an `rdf:type` triple is detected from the mappings, we try to identify the related DBpedia class. If a DBpedia class is found, all super types are assigned to a resource.

TRANSITIVE REDIRECTS DBpedia already has scripts in place to identify, extract and resolve redirects. After the redirects are extracted, a transitive redirect closure is calculated and applied in all generated datasets by replacing the redirected IRIs to the final ones.

VALIDATION The DBpedia extraction framework already takes care of the correctness of the extracted datatypes during extraction. This is achieved by making sure that the value of every property conforms to the range of that property (i.e. `xsd:date`) We provide two additional steps of validation. The first step is performed during extraction and checks if the property mappings has a compatible `rdfs:range` (literal or IRI) with the current value. The rejected triples are stored for feedback to the DBpedia mapping community. The second step is performed in a post-processing step and validates if the type of the object IRI is disjoint with the `rdfs:range` or the type of the subject disjoint with the `rdfs:domain` of the property. These inconsistent triples, although they are excluded from the SPARQL endpoint and the Linked Data interface, are offered for download. These violations may originate from logically inconsistent schema mappings or result from different schema modeling between Wikidata and DBpedia.

IRI Schemes

As mentioned earlier, we decided to generate the RDF datasets under the `wikidata.dbpedia.org` domain. For example, `wkdt:Q42` will be transformed to `dw:Q42`.

REIFICATION In contrast to Wikidata, simple RDF reification was chosen for the representation of qualifiers. This leads to a simpler design and further reuse of the DBpedia properties. The IRI schemes for the `rdf:Statement` IRIs follow the same verbose approach from DBpedia to make them easily writable manually by following a specific pattern. If the value is an IRI (Wikidata Item) then for a subject IRI Qs, a property Px and a value IRI Qv the reified statement IRI has the form `dw:Qs_Px_Qv`. If the value is a Literal then for a subject IRI Qs, a property Px and a Literal value Lv the reified statement IRI has the form `dw:Qs_Px_H(Lv,5)`, where H() is a hash function that takes as argument a string (Lv) and a number to limit the size of the returned hash (5). The hash function in the case of literals is used to create unique IRI and we consider the value '5' big enough to avoid collisions in that value space and keep it short at the same time. The equivalent representation of the Wikidata example in Section 6.1 is: [13]

```
dw:Q42_P26_Q14623681 a rdf:Statement ;
 rdf:subject dw:Q42 ;
 rdf:predicate dbo:spouse ;
 rdf:object dw:Q14623681 ;
 dbo:startDate "1991-11-25"^^xsd:date ;
 dbo:endDate "2001-5-11"^^xsd:date ;
```

Listing 7: Simple RDF reification example

IRI SPLITTING The Wikidata data model allows multiple identical claims with different qualifiers. In those not so common cases the DBw heuristic for IRI readability fails to provide unique IRIs. We create hash-based IRIs for each identical claim and attach them to the original IRI with the `dbo:wikidataSplitIri` property. At the time of writing, there are 69,662 IRI split triples and Listing 8 provides an example split IRI.

DATASET DESCRIPTION

A statistical overview of the DBw dataset is provided in Table 12. We extract provenance information, e.g. the MediaWiki page and revision IDs as well as redirects. Aliases, labels and descriptions are extracted from the related Wikidata item section and are similar to the RDF data Wikidata provides. A difference to Wikidata are the properties we chose to associate aliases and description. Each row in Table 12 is provided as a separate file to ease the consumption of parts of the DBw dataset.

Wikidata sitelinks are processed to provide three datasets: 1) `owl:sameAs` links between DBw IRIs and Wikidata IRIs (e.g. `dw:Q42 owl:sameAs`

13 DBw does not provide precision. Property definitions exist in the DBpedia ontology

```
dw:Q30_P6_Q35171 dbo:wikidataSplitIri
  dw:Q30_P6_Q35171_543e4, dw:Q30_P6_Q35171_64a6c.

dw:Q30_P6_Q35171_543e4 a rdf:Statement ;
 rdf:subject dw:Q30 ;
 rdf:predicate dbo:primeMinister ;
 rdf:object dw:Q35171 ;
 dbo:startDate "1893-3-4"^^xsd:date ;
 dbo:endDate "1897-3-4"^^xsd:date ;

dw:Q30_P6_Q35171_64a6c a rdf:Statement ;
 rdf:subject dw:Q30 ;
 rdf:predicate dbo:primeMinister ;
 rdf:object dw:Q35171 ;
 dbo:startDate "1885-3-4"^^xsd:date ;
 dbo:endDate "1889-3-4"^^xsd:date;
```

Listing 8: Example of splitting duplicate claims with different qualifiers using dbo:wikidataSplitIri

`wkdt: Q42`), 2) `owl:sameAs` links between DBw IRIs and sitelinks converted to DBpedia IRIs (e.g. `dw:Q42 owl:sameAs db-en:Douglas_Adams`) and 3) for every language in the mappings wiki we generate `owl:sameAs` links to all other languages (e.g. `db-en:Douglas_Adams owl:sameAs db-de: Douglas_Adams`). The latter is used for the DBpedia releases in order to provide links between the different DBpedia language editions.

Mapped facts are generated from the *Wikidata property mappings* (cf. Section 6.3.1). Based on a combination of the predicate and object value of a triple they are split in different datasets. Types, transitive types, geo coordinates, depictions and external `owl:sameAs` links are separated. The rest of the mapped facts are in the *mappings* dataset. The reified mapped facts (R) contains all the mapped facts as reified statements and the mapped qualifiers for these statements (RQ) are provided separately (cf. Listing 7).

Raw facts consist of three datasets that generate triples with DBw IRIs and the original Wikidata properties. The first dataset (raw facts) provides triples for simple statements. The same statements are reified in the second dataset (R) and in the third dataset (RQ) we provide qualifiers linked in the reified statements. Example of the raw datasets can be seen in Listing 7 by replacing the DBpedia properties with the original Wikidata properties. These datasets provide full coverage and, except from the reification design and different namespace, can be seen as equivalent with the WikidataRDF dumps.

Wikidata statement references are extracted in the *references* dataset using the reified statement resource IRI as subject and the `dbo:reference` property. Finally, in the mapping and ontology violation datasets we provide triples rejected according to Section 6.3.2.

Title	Triples	Description
Provenance	20.3M	PageIDs & revisions
Redirects	855K	Explicit & transitive redirects
Aliases	5.3M	Resource aliases with dbo:alias
Labels	87.1M	Labels with rdfs:label
Descriptions	137M	Descriptions with dbo:description
Sitelinks	271M	DBpedia inter-language links
Wikidata links	19.4M	Links to original Wikidata URIs
Mapped facts	320M	Aggregated mapped facts
- Types	7.9M	Direct types from the DBpedia ontology
- Transitive Types	52M	Transitive types from the DBpedia ontology
- SubClassOf	332K	Wikidata SubClasOf DBpedia ontology
- Coordinates	9.5M	Geo coordinates
- Images	2.3M	Depictions using foaf:depiction & dbo:thumbnail
- Literals	4.7M	Wikidata literals with DBpedia ontology
- Other	27.5M	Wikidata statements with DBpedia ontology
- External links	4.3M	sameAs links to external databases
Mapped facts (R)	210M	Mapped statements reified (all)
Mapped facts (RQ)	998K	Mapped qualifiers
Raw facts	82M	Raw simple statements - not mapped
Raw facts (R)	328M	Raw statements reified
Raw facts (RQ)	2.2M	Raw qualifiers
References	56.8M	Reified statements references with dbo:reference
Mapping Errors	2.9M	Facts from incorrect mappings
Ontology Violations	42K	Facts excluded due to ontology inconsistencies

Table 12: Description of the DBw datasets. (R) stands for a reified dataset and (Q) for a qualifiers dataset

Title	Before	After
Types	7,457,860	7,911,916
Transitive Types	49,438,753	52,042,144
SubClassOf	237,943	331,551

Table 13: Number of triples comparison before and after automatic class mappings extracted from Wikidata SubClassOf relations

DATASET STATISTICS

The statistics we present are based on the Wikidata XML dump from January 2016. We managed to generate a total of 1.4B triples with

Class	Count
dbo:Agent	3.43M
dbo:Person	3.08M
geo:spatialThing	2.39M
dbo:TopicalConcept	2.12M
dbo:Taxon	2.12M

Table 14: Top classes

Property	Count
owl:sameAs	320.8M
rdf:type	192.7M
dbo:description	136.9M
rdfs:label	87.2M
rdfs:seeAlso	10.1M

Table 15: Top properties

Property	Count
dbo:date	380K
dbo:startDate	265K
dbo:endDate	116K
dbo:country	40K
geo:point	31K

Table 16: Top mapped qualifiers

Property	rdfs:label	Count
wd:P31	instance of	15.3M
wd:P17	country	3.9M
wd:P21	sex or gender	2.8M
wd:P131	located in the administrative territorial entity	2.7M
wd:P625	coordinate location	2.4M

Table 17: Top properties in Wikidata

188,818,326 unique resources. In Table 12 we provide the number of triples per combined datasets.

CLASS & PROPERTY STATISTICS We provide the 5 most popular DBw classes in Table 14. We managed to extract a total of 7.9M typed Things with Agents and Person as the most frequent types. The 5 most frequent mapped properties in simple statements are provided in Table 15 and the most popular mapped properties in qualifiers in Table 16. Wikidata does not have a complete range of value types and date properties are the most frequent at the moment.

MAPPING STATISTICS In total, 269 value transformation mappings were defined along with 185 `owl:equivalentProperty` and 323 `owl:equivalentClass` schema mappings. Wikidata has 1935 properties (as of January 2016) defined with a total of 81,998,766 occurrences. With the existing mappings we covered 74.21 % of the occurrences.

REDIRECTS In the current dataset we generated 854,578 redirects – including transitive. The number of redirects in Wikidata is small compared to the project size but is is also a relatively new project. As

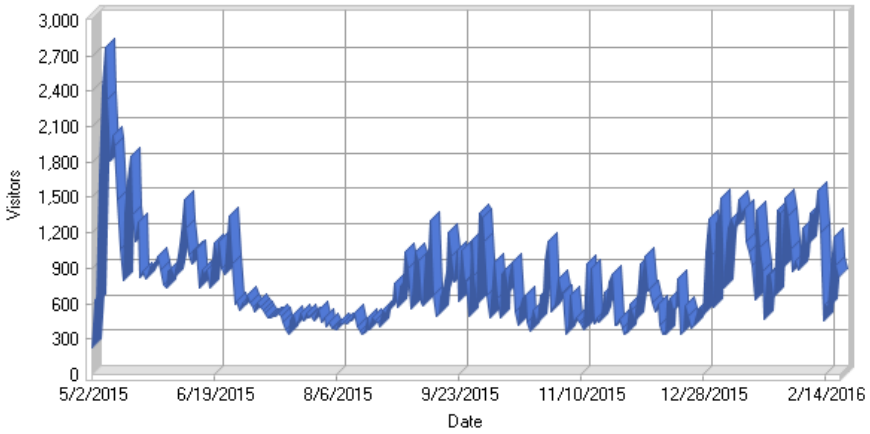


Figure 21: Number of daily visitors in http://wikidata.dbpedia.org

the project matures in time the number of redirects will increase and resolving them will have an impact on the resulting data.

VALIDATION According to Table 12, a total of 2.9M errors originated from 11 wrong Wikidata-to-DBpedia schema mappings and 42,541 triples did not pass the ontology validation (cf. Section 6.3.2).

ACCESS STATISTICS There were more than 10 million requests to wikidata.dbpedia.org since May 2015 from 28,557 unique IPs as of February 2016 and the daily visitors range from 300 to 2.7K (cf. Figure 21). The access logs were analysed by using WebLog Expert.[14] The full report can be found on our website. [15]

ACCESS AND SUSTAINABILITY

This dataset is part of the official DBpedia knowledge infrastructure and is published through the regular releases of DBpedia, along with the rest of the DBpedia language editions. The first DBpedia release that included this dataset is DBpedia release 2015-04. DBpedia is a pioneer in adopting and creating best practices for Linked Data and RDF publishing. Thus, being incorporated into the DBpedia publishing workflow guarantees: a) long-term availability through the DBpedia Association and b) agility in adopting any new best-practices promoted by DBpedia. In addition to the regular and stable releases of DBpedia we provide additional dataset updates from the project website.

Besides the stable dump availability we created `http://wikidata.dbpedia.org` for the provision of a Linked Data interface and a SPARQL Endpoint. The dataset is registered in DataHub and provides machine

14 https://www.weblogexpert.com/
15 http://wikidata.dbpedia.org/report/report.html

Table 18: Technicial details of DBw dataset

Name	DBw
Sparql Endpoint	http://wikidata.dbpedia.org/sparql
Example resource link	http://wikidata.dbpedia.org/resource/Q42
Download link	http://wikidata.dbpedia.org/downloads
DataHub link	http://datahub.io/dataset/dbpedia-wikidata
Void link	http://wikidata.dbpedia.org/downloads/void.ttl
DataID link	http://wikidata.dbpedia.org/downloads/20150330/dataid.ttl
Licence	CC0

readable metadata as void and DataID (Brümmer et al., 2014). Since the project is now part of the official DBpedia Information Extraction Framework, our dataset reuses the existing user and developer support infrastructure. DBpedia has a general discussion and developer list as well as an issue tracker[16] for submitting bugs.

USE CASES

Although it is early to identify a big range of possible use cases for DBw, our main motivation was a) familiar querying for the DBpedia community, b) vertical integration with the existing DBpedia infrastructure and c) data integration and fusion.

Listings 9 and 10 provide query examples with simple and reified statements. Since DBpedia provides transitive types directly, queries such as *select all places* can be formulated in SPARQL endpoints without SPARQL 1.1 support or simple scripts on the dumps. Moreover, `dbo:country` can be more intuitive than `wkdt:P17c`. Finally, the DBpedia queries can, in most cases directly or with minor adjustments, run on all DBpedia language endpoints. This, among others, means that existing DBpedia applications are potentially compatible with DBw. When someone is working with reified statements, the DBpedia IRIs encode all possible information to visually identify the resources and items involved (cf. Section 6.3.3) in the statement while Wikidata uses a hash string. Querying for reified statement in Wikidata needs to properly suffix the Wikidata property with `c/s/q`. [17] Simple RDF

16 https://github.com/dbpedia/extraction-framework/issues

17 At the time of writing, there is a mismatch of the actual Wikidata syntax reported from the Wikidata paper, the Wikidata RDF dumps and the official Wikidata SPARQL endpoint

reification on the other hand limits the use of SPARQL property path expressions.

```
#DBw
SELECT * WHERE {
 ?place a dbo:Place ;
        dbo:country dw:Q183.
 OPTIONAL {
    ?place rdfs:label ?label.
    FILTER (LANG(?label)="en")
 }
}

#Wikidata
SELECT * WHERE {
  ?place wdt:P31/wdt:P279* wd:Q486972;
         wdt:P17 wd:Q183.
  OPTIONAL {
    ?place rdfs:label ?label.
    FILTER (LANG(?label)="en")
  }
}
```

Listing 9: Queries with simple statement

```
#DBw
SELECT DISTINCT ?person WHERE {
 ?statementUri rdf:subject ?person ;
               rdf:predicate dbo:spouse ;
               dbo:startDate ?m_date.
 FILTER (year(?m_date)<2000)
}

#Wikidata
SELECT DISTINCT ?person WHERE {
 ?person p:P26/pq:P580 ?m_date.
 FILTER (year(?m_date)<2000)
}
```

Listing 10: Queries with reified statements

The fact that the datasets are split according to the information they contain eases data consumption when someone needs a specific subset, e.g. coordinates. An additional important use case is data integration. Converting a dataset to a common schema, facilitates the integration of data. The DBw dataset is planned to be used as an enrichment dataset on top of DBpedia and fill in data that are being moved from Wikipedia infoboxes to Wikidata. It is also part of our short-term plan to fuse all DBpedia data into a new single knowledge base and the DBw dataset will have a prominent role in this project.

USE CASES FOR WIKIDATA Through DBw Wikidata has a gateway to DBpedia data from other language editions by using the *Wikidata*

property mappings. By accessing DBpedia data, Wikidata can cross-reference facts as well as identify & consume data updates from Wikipedia. Another core feature of Wikidata is adding references for each statement. Unfortunately there are many facts copied from Wikipedia by Wikidata editors that cite Wikipedia and not the possible external or authoritative source that is cited in the Wikipedia article. DBpedia recently started extracting citations from Wikipedia pages. This makes it possible to associate facts extracted from DBpedia with citations close to the fact position. By using the DBpedia citations, DBpedia facts and their associations as well as the *Wikidata property mappings*, a lot of references with high confidence can be suggested for Wikidata facts.

COMBINATION OF BOTH DATASETS Currently there is indeed an overlap of facts that exist both in DBpedia and Wikidata however, there are also a lot of facts that are unique to each dataset. For instance, DBpedia captures the links between Wikipedia pages that are used to compute page-rank datasets for different Wikipedia/DBpedia language editions. Using the page links from DBpedia and Wikidata as an article association hub, a global page-rank score for Wikidata items that takes the interconnection graph of all Wikipedias is possible. In general DBw provides a bridge that we hope will make it easier for the huge amount of information on both datasets to be used in some new interesting ways and improve Wikidata and Wikipedia itself. [18]

CONCLUSIONS AND FUTURE WORK

We present an effort to provide an alternative RDF representation of Wikidata. Our work involved the creation of 10 new DBpedia extractors, a Wikidata2DBpedia mapping language and additional post-processing & validation steps. With the current mapping status we managed to generate over 1.4 billion RDF triples with CC0 license. According to the web server statistics, the daily number of DBw visitors range from 300 to 2,700 and we counted almost 30,000 unique IPs since the start of the project, which indicates that this dataset is used. In the future we plan to extend the mapping coverage as well as extend the language with new mapping functions and more advanced mapping definitions. The dataset is already part of the bi-yearly DBpedia release cycle and thus regularly updated. We will additionally consider providing DBw as a live service similar to DBpedia Live.

18 http://wiki.dbpedia.org/about

7

RELATED WORK

Lehmann, Isele, Jakob, Jentzsch, Kontokostas, Mendes, Hellmann, Morsey, Kleef, Auer, and Bizer, (2015)

CROSS DOMAIN COMMUNITY KNOWLEDGE BASES

Wikidata

In March 2012, the Wikimedia Germany e.V. started the development of Wikidata[1]. Wikidata is a free knowledge base about the world that can be read and edited by humans and machines alike. It provides data in all languages of the Wikimedia projects, and allows for central access to the data in a similar vein as Wikimedia Commons does for multimedia files. Things described in the Wikidata knowledge base are called items and can have labels, descriptions and aliases in all languages. Wikidata does not aim at offering a single truth about things, but providing statements given in a particular context. Rather than stating that Berlin has a population of 3.5 million, Wikidata contains the statement about Berlin's population being 3.5 million as of 2011 according to the German statistical office. Thus, Wikidata can offer a variety of statements from different sources and dates. As there are potentially many different statements for a given item and property, ranks can be added to statements to define their status (preferred, normal or deprecated).

In particular, Wikidata is a collection of entity pages and there are two types of entity pages: items and properties. Every item page contains labels, short description, aliases, statements and site links. As depicted in Figure 22, each statement consists of a claim and one or more optional references. Each claim consists of a property-value pair, and optional qualifiers. Values are also divided into three types: no value, unknown value and custom value. The "no value" marker means that there is certainly no value for the property, the "unknown value" marker means that the property has some value, but it is unknown to us and the "custom value " which provides a known value for the property.

The initial development was divided in three phases:

- The first phase (interwiki links) created an entity base for the Wikimedia projects. This provides a better alternative to the previous interlanguage link system.
- The second phase (infoboxes) gathered infobox-related data for a subset of the entities, with the explicit goal of augmenting

1 http://wikidata.org/

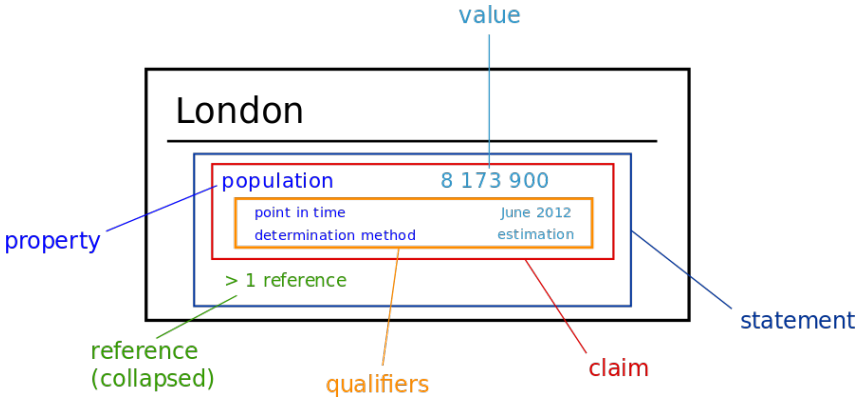


Figure 22: Wikidata statements, image taken from Commons

```
wkdt:Q42 wkdt:P26s wkdt:Q42Sb88670f8-456b-3ecb-cf3d-2bca2cf7371e.
wkdt:Q42Sb88670f8-456b-3ecb-cf3d-2bca2cf7371e wkdt:P580q wkdt:VT74cee544.
wkdt:VT74cee544 rdf:type :TimeValue.;
 :time "1991-11-25"^^xsd:date;
 :timePrecision "11"^^xsd:int; :preferredCalendar wkdt:Q1985727.
wkdt:Q42Sb88670f8-456b-3ecb-cf3d-2bca2cf7371e wkdt:P582q wkdt:VT162aadcb.
wkdt:VT162aadcb rdf:type :TimeValue;
 :time "2001-5-11"^^xsd:date;
 :timePrecision "11"^^xsd:int; :preferredCalendar wkdt:Q1985727.
```

Listing 11: RDF serialization for the fact: Douglas Adams' (Q42) spouse is Jane Belson (Q14623681) from (P580) 25 November 1991 until (P582) 11 May 2001. Extracted from (Vrandečić and Krötzsch, 2014) Figure 3

the infoboxes that are currently widely used with data from Wikidata.

- The third phase (lists) will expand the set of properties beyond those related to infoboxes, and will provide ways of exploiting this data within and outside the Wikimedia projects.

Since March 2013 the Wikidata extension is live on all Wikipedia language editions and thus their pages can be linked to items in Wikidata and include data from Wikidata. At the time of writing of this article, the development of the third phase is ongoing.

As of April 2016, Wikidata contains more than 20 million items and 87 million statements[2] and has more than 6,000 active users.[3] In 2014, an RDF export of Wikidata was introduced (Erxleben et al., 2014) and recently a few SPARQL endpoints were made available as external contributions as well as an official one later on.[4]. Listing 11 provides an example RDF serialization of a Wikidata statement.

2 https://tools.wmflabs.org/wikidata-todo/stats.php
3 https://stats.wikimedia.org/wikispecial/EN/TablesWikipediaWIKIDATA.htm
4 https://query.wikidata.org/

Freebase

Freebase[5] is a graph database, which also extracts structured data from Wikipedia and makes it available in RDF. Both DBpedia and Freebase link to each other and provide identifiers based on those for Wikipedia articles. They both provide dumps of the extracted data, as well as APIs or endpoints to access the data and allow their communities to influence the schema of the data. There are, however, also major differences between both projects. DBpedia focuses on being an RDF representation of Wikipedia and serving as a hub on the Web of Data, whereas Freebase uses several sources to provide broad coverage. The store behind Freebase is the GraphD (Meyer et al., 2010) graph database, which allows to efficiently store metadata for each fact. This graph store is append-only. Deleted triples are marked and the system can easily revert to a previous version. This is necessary, since Freebase data can be directly edited by users, whereas information in DBpedia can only indirectly be edited by modifying the content of Wikipedia or the Mappings Wiki. From an organisational point of view, Freebase is mainly run by Google, whereas DBpedia is an open community project. In particular in focus areas of Google and areas in which Freebase includes other data sources, the Freebase database provides a higher coverage than DBpedia.

YAGO

One of the projects that pursues similar goals to DBpedia is *YAGO*[6] (Suchanek, Kasneci, and Weikum, 2007). YAGO is identical to DBpedia in that each article in Wikipedia becomes an entity in YAGO. Based on this, it uses the leaf categories in the Wikipedia category graph to infer type information about an entity. One of its key features is to link this type information to WordNet. WordNet synsets are represented as classes and the extracted types of entities may become subclasses of such a synset. In the *YAGO2 system* (Hoffart et al., 2013a), declarative extraction rules were introduced, which can extract facts from different parts of Wikipedia articles, e.g. infoboxes and categories, as well as other sources. YAGO2 also supports spatial and temporal dimensions for facts at the core of its system.

One of the main differences between DBpedia and YAGO in general is that DBpedia tries to stay very close to Wikipedia and provide an RDF version of its content. YAGO focuses on extracting a smaller number of relations compared to DBpedia to achieve very high precision and consistent knowledge. The two knowledge bases offer different type systems: whereas the DBpedia ontology is manually maintained, YAGO is backed by WordNet and Wikipedia leaf categories. Due to this, YAGO contains many more classes than DBpe-

5 http://www.freebase.com/
6 http://www.mpi-inf.mpg.de/yago-naga/yago/

dia. Another difference is that the integration of attributes and objects in infoboxes is done via mappings in DBpedia and, therefore, by the DBpedia community itself, whereas this task is facilitated by expert-designed declarative rules in YAGO2.

The two knowledge bases are connected, e.g. DBpedia offers the YAGO type hierarchy as an alternative to the DBpedia ontology and `sameAs` links are provided in both directions. While the underlying systems are very different, both projects share similar aims and positively complement and influence each other.

KNOWLEDGE EXTRACTION FROM WIKIPEDIA

Since its official start in 2001, Wikipedia has always been the target of automatic extraction of information due to its easy availability, open license and encyclopedic knowledge. A large number of parsers, scraper projects and publications exist. In this section, we restrict ourselves to approaches that are either notable, recent or pertinent to DBpedia. MediaWiki.org maintains an up-to-date list of software projects[7], who are able to process wiki syntax, as well as a list of data extraction extensions[8] for MediaWiki.

JWPL (Java Wikipedia Library, (Zesch, Müller, and Gurevych, 2008)) is an open-source, Java-based API that allows to access information provided by the Wikipedia API (redirects, categories, articles and link structure). JWPL contains a MediaWiki Markup parser that can be used to further analyze the contents of a Wikipedia page. Data is also provided as XML dump and is incorporated in the lexical resource UBY[9] for language tools.

Several different approaches to extract knowledge from Wikipedia are presented in (Nakayama et al., 2008). Given features like anchor texts, interlanguage links, category links and redirect pages are utilized e.g. for word-sense disambiguations or synonyms, translations, taxonomic relations and abbreviation or hypernym resolution, respectively. Apart from this, link structures are used to build the *Wikipedia Thesaurus* Web service[10]. Additional projects that exploit the mentioned features are listed on the *Special Interest Group on Wikipedia Mining (SIGWP)* Web site[11].

An earlier approach to improve the quality of the infobox schemata and contents is described in (Wu and Weld, 2007). The presented methodology encompasses a three step process of *preprocessing*, *classification* and *extraction*. During preprocessing refined target infobox schemata are created applying statistical methods and training sets are extracted based on real Wikipedia data. After assigning a class

7 http://www.mediawiki.org/wiki/Alternative_parsers
8 http://www.mediawiki.org/wiki/Extension_Matrix/data_extraction
9 http://www.ukp.tu-darmstadt.de/data/lexical-resources/uby/
10 http://sigwp.org/en/index.php/Wikipedia_Thesaurus
11 http://sigwp.org/en/

and the corresponding target schema (classification) the training sets are used to extract target infobox values from the document's text applying machine learning algorithms.

The idea of using structured data from certain markup structures was also applied to other user-driven Web encyclopedias. In (Niu et al., 2011) the authors describe their effort building an integrated *Chinese Linking Open Data (CLOD)* source based on the Chinese Wikipedia and the two widely used and large encyclopedias *Baidu Baike*[12] and *Hudong Baike*[13]. Apart from utilizing MediaWiki and HTML Markup for the actual extraction, the Wikipedia interlanguage links were used to link the CLOD source to the English DBpedia.

A more generic approach to achieve a better cross-lingual knowledge-linkage beyond the use of Wikipedia interlanguage links is presented in (Wang et al., 2012). Focusing on wiki knowledge bases the authors introduce their solution based on structural properties like similar linkage structures, the assignment to similar categories and similar interests of the authors of wiki documents in the considered languages. Since this approach is language-feature-agnostic it is not restricted to certain languages.

KnowItAll[14] is a web scale knowledge extraction effort, which is domain-independent, and uses generic extraction rules, co-occurrence statistics and Naive Bayes classification (Etzioni et al., 2004). Cyc (Lenat, 1995) is a large common sense knowledge base, which is now partially released as OpenCyc and also available as an OWL ontology. OpenCyc is linked to DBpedia, which provides an ontological embedding in its comprehensive structures. WikiTaxonomy (Ponzetto and Strube, 2008) is a large taxonomy derived from categories in Wikipedia by classifying categories as instances or classes and deriving a subsumption hierarchy. The KOG system (Wu and Weld, 2008) refines existing Wikipedia infoboxes based on machine learning techniques using both SVMs and a more powerful joint-inference approach expressed in Markov Logic Networks. KYLIN (Wu and Weld, 2007) is a system which autonomously extracts structured data from Wikipedia and uses self-supervised linking.

12 http://baike.baidu.com/
13 http://www.hudong.com/
14 http://www.cs.washington.edu/research/knowitall/

Part III

RDF AND LINKED DATA QUALITY ASSESSMENT

8

TEST-DRIVEN QUALITY ASSESSMENT METHODOLOGY

Kontokostas, Brümmer, Hellmann, Lehmann, and Ioannidis, (2014b,c)

Linked Open Data (LOD) comprises an unprecedented volume of structured data published on the Web. However, these datasets are of varying quality ranging from extensively curated datasets to crowdsourced and even extracted data of relatively low quality. Data quality is not an absolute measure, but assesses fitness for use (Juran, 1974). Consequently, one of the main challenges regarding the wider deployment and use of semantic technologies on the Web is the assessment and ensuring of the quality of a certain, possibly evolving, dataset for a particular use case. There have been few approaches for assessing Linked Data quality. However, these were majorly methodologies, which require (1) a large amount of manual configuration and interaction (Bizer and Cyganiak, 2009; Flemming, 2010; Mendes P.N., 2012) or (2) automated, reasoning based methods (Guéret et al., 2012; Ji et al., 2009). While reasoning based methods allow more automation, they are either limited to very specific quality aspects (such as link quality (Guéret et al., 2012)) or lack scalability to the medium and large datasets being increasingly published as Linked Data. In consequence, we observe a shortage of practical quality assessment approaches for Linked Data, which balance between a high degree of automation *and* scalability to datasets comprising billions of triples.

In this chapter, we present a methodology for test-driven Linked Data quality assessment, which is inspired by test-driven software development. In software engineering, a *test case* can be defined as "*an input on which the program under test is executed during testing*" and a *test set* as "*a set of test cases for testing a program*" (Zhu, Hall, and May, 1997). A basic metric in software unit-testing is *test adequacy*, which measures the completeness of the test set. 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. In this work, we exploit the RDF data model by devising a pattern-based approach for the data quality tests of RDF knowledge bases. We argue that ontologies, vocabularies and knowledge bases should be accompanied by a number of test cases, which help to ensure a basic level of quality.

We present a methodology for assessing the quality of linked data resources, based on a formalization of data quality integrity constraints. Our formalization employs SPARQL query templates, which are instantiated into concrete quality test case queries. Based on an extensive survey, we compile a comprehensive library of quality test case patterns, which can be instantiated for rapid development of more test cases. We provide a method for automatic test case instantiation from these patterns for a particular ontology or vocabulary schema. Furthermore, we support the automatic derivation from OWL schema axioms. Since many schemata of LOD datasets are not very expressive, our methodology also includes semi-automatic schema enrichment. Concrete test cases are equipped with persistent identifiers to facilitate test tracking over time. We devise the notion of RDF test case coverage based on a combination of six individual coverage metrics (four for properties and two for classes).

As a result, the test case coverage can be explicitly stated for a certain dataset and potential users can thus obtain a more realistic assessment of the quality they can expect. Since the test cases are related to certain parts of the knowledge base (i.e. properties and classes), the quality of particular fragments relevant for a certain use-case can also be easily assessed. Another benefit of test-driven data engineering is support for data evolution. 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.

Our approach allows to perform an automatic test case instantiation based on schema constraints or semi-automatically enriched schemata and allows users to generate specific test case instantiations that are applicable for a schema or a dataset.

The remainder of this chapter article is structured as follows: Section 8.1 describes the methodology we followed to define *Data Quality Test Case Patterns*. We provide the basic notions of the methodology, our proposed workflow, test coverage metrics as well as discussion on how our approach compared to OWL reasoning. In Section 8.2 we provide the *test-driven data engineering* ontology that provides a port of our methodology to OWL. The elicitation of our pattern library is described in Section 8.3 and we conclude in Section 8.4.

TEST-DRIVEN DATA QUALITY METHODOLOGY

We first introduce the basic notions in our methodology, then describe its workflow and finally define test case coverage criteria analogous to unit tests in software engineering.

Basic Notions

Data Quality Test Pattern (DQTP). A data quality test case pattern is a tuple (V, S), where V is a set of typed pattern variables and S is a SPARQL query template with placeholders for the variables from V. Possible types of the pattern variables are IRIs, literals, operators, datatype values (e.g. integers) and regular expressions. With $R(v)$ we denote the value range for a pattern variable in $v \in V$, i.e. the set of values by which the variable can be substituted, and with $R(V)$ the union of all these sets, i.e. $R(V) = \bigcup_{v \in V} R(v)$.

Ideally, DQTPs should be knowledge base and vocabulary agnostic. Using %%v%% as syntax for placeholders, an example DQTP is:

```
SELECT ?s WHERE { ?s  %%P1%%   ?v1 .
                  ?s  %%P2%%   ?v2 .
                  FILTER ( ?v1 %%OP%% ?v2 ) }
```

This DQTP can be used for testing whether a value comparison of two properties P1 and P2 holds with respect to an operator OP. DQTPs represent abstract patterns, which can be further refined into concrete data quality test cases using test pattern bindings.

Test Case Pattern Binding. A test case pattern binding is a specific instantiation of a DQTP. It is a triple (σ, S, C) in which $\sigma : V \to R(V)$ is a mapping of variables to valid replacements, S is a SPARQL query template and $C \in \{error, bad\_smell\}$ is used as classification of the error.

Data Quality Test Cases. Applying σ to S results in a SPARQL query, which can then be executed. Each result of the query is considered to be a violation of a unit test. An example test case pattern binding and resulting data quality test case is[1]:

```
P1  =>  dbo:birthDate |   SELECT ?s WHERE {
P2  =>  dbo:deathDate |    ?s  dbo:birthDate  ?v1.
OP  =>  >             |    ?s  dbo:deathDate  ?v2.
                      |    FILTER ( ?v1 > ?v2 ) }
```

A test case has four different results: success (empty result), violation (results are returned), timeout (test case is marked for further inspection) and error (the query cannot be evaluated due to e.g. a network error or SPARQL engine limitations).

Test case Auto Generators (TAG). Many knowledge bases use RDFS and OWL as modelling languages. While the core of those languages aims at inferring new facts, a number of constructs is also suitable for verifying data quality. In previous work, tools like the *Pellet Integrity Constraint Validator* (Sirin and Tao, 2009) made use of this by viewing OWL axioms as constraints and reporting violations of them. Those are then interpreted via integrity constraint semantics, which uses a

1 We use http://prefix.cc to resolve all name spaces and prefixes. A full list can be found at http://prefix.cc/popular/all

closed world assumption and a weaker form of the unique names assumption in which two individuals are considered to be different unless they are explicitly stated to be equal. We pursue the same approach for re-using schema information in our test framework. To achieve this, a test case auto generator (TAG) takes a schema as input and returns test cases. We provide support for the following OWL constructs `rdfs:domain`, `rdfs:range`, `owl:minCardinality`, `owl:maxCardinality`, `owl:cardinality`, `owl:functionalProperty`, `owl:disjointClass`, `owl:propertyDisjointWith`, `owl:complementOf`, `owl:InverseFunctionalProperty`, `owl:AsymmetricProperty`, `owl:IrreflexiveProperty` and `owl:deprecated`.

Generators consist of a detection and an execution part. The detection part is a query against a schema, for instance:

```
1 SELECT DISTINCT ?T1 ?T2 WHERE {
2   ?T1 owl:disjointWith ?T2 . }
```

For every result of a detection query, a test case is instantiated from the respective pattern, for instance:

```
1 SELECT DISTINCT ?s WHERE {
2   ?s rdf:type %%T1%% .
3   ?s rdf:type %%T2%% .}
```

Depending on the violation, there is not necessarily a one-to-one mapping between a detection query and the generated test cases. For the `owl:cardinality` constraint, for example, we use three TAGs: (i) a TAG for the case a cardinality is 0, which checks whether the corresponding triple pattern is instantiated and two generators for values greater than 0, (ii) one to ensure that the property exists (TYPRODEP) and (iii) a second to validate the property occurrences (OWLCARD). The detection queries can be quite complex, we would like to stress, however, that our goal is not to provide complete reasoning and constraint checking, but rather a lightweight mechanism verifying typical violations efficiently.

Workflow

Our methodology is illustrated in Figure 23. As shown in the figure, there are two major sources for creating test cases. One source is stakeholder feedback from everyone involved in the usage of a dataset and the other source is the already existing RDFS/OWL schema of a dataset. Based on this, there are several ways in which test cases can be created:

1. *Using RDFS/OWL constraints directly:* As previously explained, test cases can be automatically created via TAGs in this case.

2. *Enriching the RDFS/OWL constraints:* Since many datasets provide only limited schema information, we perform automatic

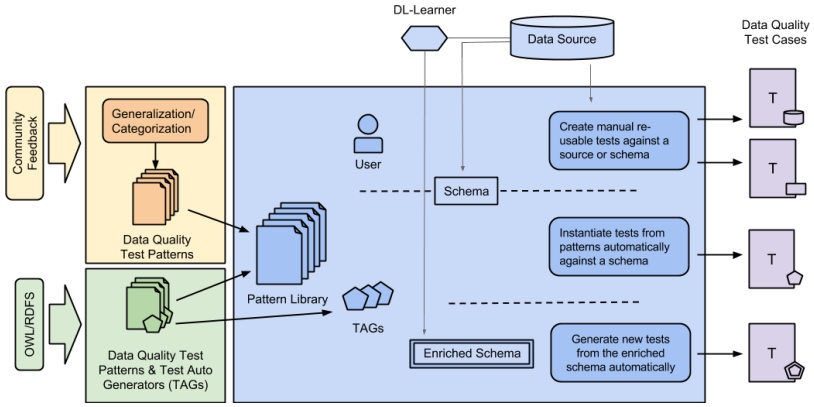


Figure 23: Flowchart showing the test-driven data quality methodology. The left part displays the input sources of our pattern library. In the middle part the different ways of pattern instantiation are shown which lead to the Data Quality Test Cases on the right.

schema enrichment as recently studied in (Bühmann and Lehmann, 2012, 2013). These schema enrichment methods can take an RDF/OWL dataset or a SPARQL endpoint as input and automatically suggest schema axioms with a certain confidence value by analysing the dataset. In our methodology, this is used to create further test cases via TAGs. It should be noted that test cases are explicitly labelled, such that the engineer knows that they are less reliable than manual test cases.

3. *Re-using tests based on common vocabularies:* Naturally, a major goal in the Semantic Web is to re-use existing vocabularies instead of creating them from scratch for each dataset. We detect the used vocabularies in a dataset, which allows to re-use test cases from a test case pattern library. The creation of that library is described in the next section.

4. *Instantiate existing DQTPs:* The aim of DQTPs is to be generic, such that they can be applied to different datasets. While this requires a high initial effort of compiling a pattern library, it is beneficial in the long run, since they can be re-used. Instead of writing SPARQL templates themselves, an engineer can select and instantiate the correct DQTP. This does not necessarily require SPARQL knowledge, but can also be achieved via a textual description of a DQTP, examples and its intended usage.

5. *Write own DQTPs:* In some cases, test cases cannot be generated by any of the automatic and semi-automatic methods above and have to be written from scratch by an engineer. These DQTPs can then become part of a central library to facilitate later re-use.

RDF DATA SOURCE Our methodology encapsulates all RDF data as *Sources*. An RDF data source can be a schema, a dataset, a semi-

automatically enriched schema (based on a dataset). Test cases can be associated with any *Source* and can be generated either automatically or manually. A source can also be related to other sources. For example a dataset can reference many schema sources, or a schema may re-use other schemas.

Test Coverage and Adequacy

In software engineering, a *test case* can be defined as *an input on which the program under test is executed during testing* and a *test set* as *a set of test cases for testing a program* (Zhu, Hall, and May, 1997). A basic metric in software unit testing is *Test Adequacy*. According to (Zhu, Hall, and May, 1997), adequacy is a notion that measures the completeness of the test set. An *Adequacy Stopping Rule* (ASR) is a related metric with a range {true|false} that defines whether sufficient testing has been done. Many attempts have been made to quantify test adequacy with the main coverage criteria being: a) statement coverage, b) branch coverage, c) path coverage and d) mutation adequacy. It is hard to automate the creation of these tests.

In RDF, instead of code, the testing subject is *data* that is stored in triples and adheres to a schema. We define an RDF test case as *a data constraint that involves one or more triples* and an RDF test set as *a set of test cases for testing a dataset*. Since no branches and paths in RDF exist, a *test adequacy* metric can only be related to the selectivity of the test cases. We will subsequently consider coverage as a composite of the following coverage criteria:

- *Property domain coverage (dom):* Identifies the ratio of property occurrences, where a test case is defined for verifying domain restrictions of the property.
- *Property range coverage (ran):* Identifies the ratio of property occurrences, where a test case is defined for verifying range restrictions of the property.
- *Property dependency coverage (pdep):* Identifies the ratio of property occurrences, where a test case is defined for verifying dependencies with other properties.
- *Property cardinality coverage (card):* Identifies the ratio of property occurrences, where a test case is defined for verifying the cardinality of the property.
- *Class instance coverage (mem):* Identifies the ratio of classes with test cases regarding class membership.
- *Class dependency coverage (cdep):* Identifies the ratio of class occurrences for which test cases verifying relationships with other classes are defined.

A certain property should also be considered to be covered, if the absence of a particular constraint is explicitly stated.

The above criteria can be computed by *coverage computation functions*. Each coverage computation function $f : Q \rightarrow 2^E$ takes a SPARQL query $q \in Q$ corresponding to a test case pattern binding as input and returns a set of entities. As an example, the function f_{dom} for computing the domain coverage returns the set of all properties p such that the triple pattern $(?s, p, ?o)$ occurs in q and there is at least one other triple pattern using $?s$ in q. This can straightforwardly be extended to a function $F : 2^Q \rightarrow 2^E$ taking a set of SPARQL queries as input and returning a set of entities. F computes how many entities are covered by the test case queries. For properties, F can be further extended to a function F' with $F'(QS, D) = \sum_{p \in F(QS)} pfreq(p)$ where $pfreq(p)$ is the frequency of a property p, i.e. the number of occurrences of p divided by the number of occurrences of all properties in D. The extension for classes is analogous. This extension weights the entities by their frequency in the dataset. We propose to employ occurrences, i.e. concrete entity usages, instead of properties itself in order to reduce the influence of rarely used properties on the coverage.

The other coverage criteria are defined as follows: Range coverage f_{ran} is analogous to domain coverage. The property dependency coverage f_{pdep} of a query q returns all properties in q if there are at least two different properties and an empty set otherwise. Property cardinality coverage f_{card} of a query q returns the set of all properties p, such that $(?s, p, ?o)$ occurs in q along with `GROUP BY ?s` as well as `HAVING(count(?s) op n)` aggregates (op is one of $\leqslant, <, =, >, \geqslant$ and n a number) or, analogously, the same criteria for $?o$ instead of $?s$. Class instance coverage f_{mem} of a query q returns the set of all classes c such that $(?s, \textit{rdf:type}, c)$ occurs in q. The class dependency coverage f_{cdep} of a query q returns all classes in q if there are at least two different classes and an empty set otherwise.

In the above definition, please note that domain and range restrictions are more general than verifying `rdfs:domain` and `rdfs:range` as they cover all test cases, which can be performed via SPARQL on subject and objects of triples using a particular property. Please note that many test cases can be expressed in OWL 2, in particular when using the Pellet integrity constraint semantics. For instance, custom datatypes in OWL2[2] can be used for range checking of property values using regular expressions. As noted above, we transparently support the usage of OWL, but some test cases are much easier to implement in SPARQL and others, e.g. the SKOS restriction: "A resource has no more than one value of `skos:prefLabel` per language tag" cannot be checked in OWL at all, but is a straightforward DQTP in our case (ONELANG in Table 19).

2 http://www.w3.org/TR/owl2-primer/#Advanced_Use_of_Datatypes

Formally, we can define RDF test case coverage Cov of a set of test case queries QS with respect to a dataset D as follows:

$$\begin{aligned} Cov(QS, D) = \frac{1}{6}(&F'_{dom}(QS, D) + F'_{ran}(QS, D) \\ &+ F'_{pdep}(QS, D) + F'_{card}(QS, D) \\ &+ F'_{mem}(QS, D) + F'_{cdep}(QS, D)) \end{aligned}$$

The coverage is a heuristic in the $[0, 1]$ range, which helps to assess whether the defined test cases are sufficient for data quality assessment. Higher results represent better coverage.

Relation to OWL Reasoning.

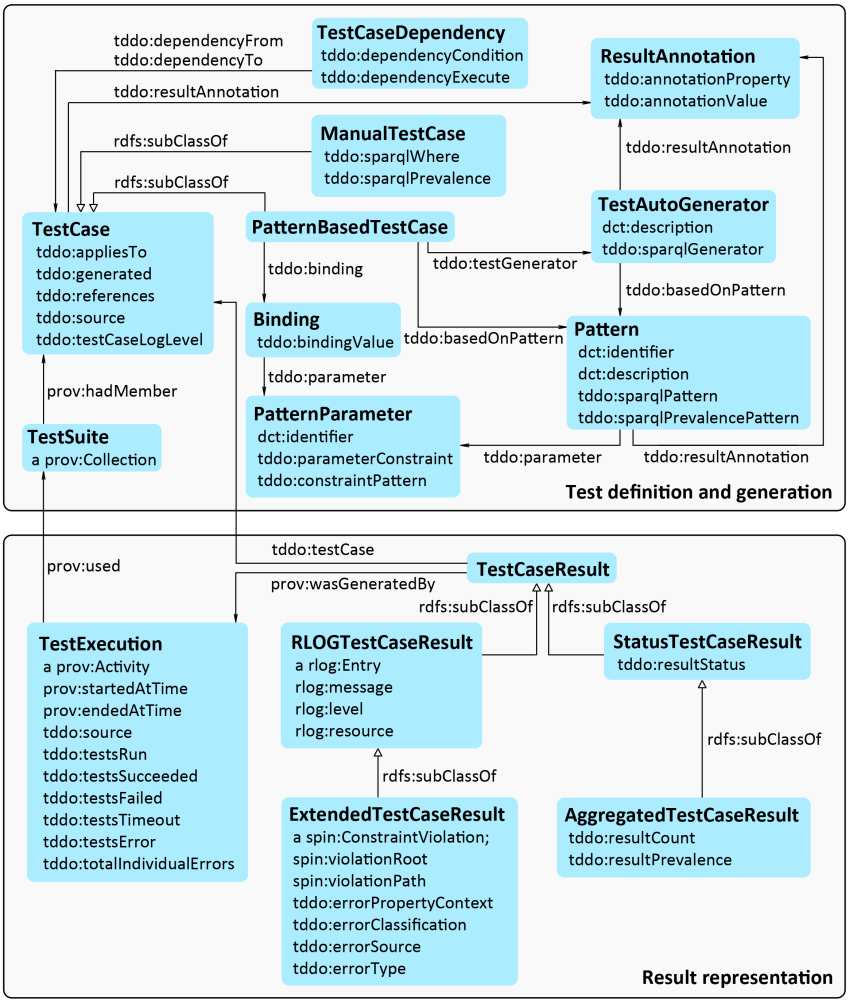
SPARQL test cases can detect a subset of common validation errors detectable by a sound and complete OWL reasoner. However, this is limited by a) the reasoning support offered by the used SPARQL endpoint and b) the limitations of the OWL-to-SPARQL translation. On the other hand, SPARQL test cases can find validation errors that are not expressible in OWL, but within the expressivity of SPARQL (see Angles and Gutierrez, (2008)) for more details and a proof that SPARQL 1.0 has the same expressive power as relational algebra under bag semantics). This includes aggregates, property paths, filter expressions etc. Please note that for scalability reasons full OWL reasoning is often not feasible on large datasets. Furthermore, many datasets are already deployed and easy to access via SPARQL endpoints. Additionally, the *Data Quality Test Pattern* (DQTP) library may arguably provide a more user friendly approach for building validation rules compared to modelling OWL axioms. However, the predefined DQTP library has some limitations as well, in particular a) it requires familiarity with the library in order to choose the correct DQTP and 2) custom validations cannot always correspond to an existing DQTP and manual SPARQL test cases are required.

TEST DRIVEN DATA ENGINEERING ONTOLOGY

The Test Driven Data Assessment methodology is implemented using RDF as input and output and complies with our accompanied ontology.[3] The ontology additionally serves as a self-validation layer for the application input (test-cases, DQTPs and TAGs) and output (validation results). The ontology consists of 20 classes and 36 properties and reuses the PROV (Belhajjame et al., 2013), RLOG[4] and SPIN (Knublauch, Hendler, and Idehen, 2011). As depicted in Figure 24, the ontology is centered around two concepts, the test case definition and generation and the result representation.

3 http://RDFUnit.aksw.org/ns/core#

4 http://persistence.uni-leipzig.org/nlp2rdf/ontologies/rlog#



Prefix prov: http://www.w3.org/ns/prov#
Prefix spin: http://spinrdf.org/spin#
Prefix dct: http://purl.org/dc/terms/
Prefix tddo: http://databugger.aksw.org/ns/core#
Prefix rlog: http://persistence.uni-leipzig.org/nlp2rdf/ontologies/rlog#

Figure 24: Class dependencies for the test driven data engineering ontology.

Test case definition and generation. We encapsulate a list of test cases in a *TestSuite*, a subclass of `prov:Collection` that enumerates the contained test cases with `prov:hadMember`. The class *TestCase* describes an abstract test case. For each test case, we provide provenance with the following properties:

- `:appliesTo` to denote whether the test case applies to a schema, a dataset or an application.
- `:source`, the URI of the schema, dataset or application.
- `:generated` on how the test case was created (automatic or manually).
- `:references` a list of URIs a test case uses for validation.
- `:testCaseLogLevel` an `rlog:Level` this test case is associated with. In accordance to software development, the available log levels are: TRACE, DEBUG, INFO, WARN, ERROR and FATAL.

Additionally, each *TestCase* is associated with two SPARQL queries, a query for the constraint violations and a query for the prevalence of the violations. The prevalence query is optional because it cannot be computed in all cases.

```
# Violation Query            | # Prevalence Query
SELECT DISTINCT ?s WHERE {   | select count(distinct ?s) WHERE {
  ?s  dbo:birthDate  ?v1.    |   ?s dbo:birthDate ?v1 .
  ?s  dbo:deathDate  ?v2.    |   ?s dbo:deathDate ?v2 . }
  FILTER ( ?v1 > ?v2 ) }     |
```

Concrete instantiations of a *TestCase* are the *ManualTestCase* and the *PatternBasedTestCase* classes. In the former, the tester defines the SPARQL queries manually while the in the latter she provides *Bindings* for a *Pattern*. Additionally, the ontology allows the definition of dependencies between test cases. For example *if test case A fails, do not execute test case B*. This is achieved with the *TestCaseDependency* class where `:dependencyFrom` and `:dependencyTo` define the dependent test cases, `:dependencyCondition` is the status result that triggers an execute or don't execute (`:dependencyExecute`) for the dependant test case.

A *Pattern* is identified and described with the `dct:identifier` and `dct:description` properties. The `:sparqlPattern` and `:sparqlPrevalencePattern` properties hold the respective SPARQL queries with placeholder for replacement. For each placeholder a *PatternParameter* is defined and connected to the pattern with the `:parameter` property.

PatternParameters are described with a `dct:identifier` and two restriction properties: the `:parameterConstraint` to restrict the type of a parameter to Operator, Resource, Property or Class and the optional `:constraintPattern` for a regular expression constraint on the parameter values.

Bindings link to a *PatternParameter* and a value through the `:parameter` and `:bindingValue` properties respectively. *PatternBasedTestCases* are associated with *Bindings* through the `:binding` property.

```
[] a tddo:PatternBasedTestCase ;
   tddo:binding [ a       tddo:Binding ;
      tddo:bindingValue lemon:Node ;
      tddo:parameter tddp:OWLDISJC-T1 ] ;
```

A *PatternBasedTestCase* can be automatically instantiated through a *TestAutoGenerator*. Generators hold a `dct:description`, a sparql query (`:generatorSparql`) and a link to a pattern (`:basedOnPattern`).

Result representation. For the result representation we reuse the PROV Ontology. The *TestExecution* class is a subclass of `prov:Activity` that executes a *TestSuite* (`prov:used`) against a `:source` and generates a number of *TestCaseResults*. Additional properties of the *TestExecution* class are `prov:startedAtTime` and `prov:endedAtTime` as well as aggregated execution statistics like: `:testsRun`, `:testsSucceeded`,

`:testsFailed`, `:testsTimeout`, `:testsError` and `:totalIndividualErrors`.

The ontology supports four levels or result reporting, two for report on the test case level and two for individual error reporting. All result types are subclasses of the *TestCaseResult* class and for provenance we link to a *TestCase* with `:testCase` and a *TestExecution* with `prov:wasGeneratedBy` properties. The *StatusTestCaseResult* class contains a single `:resultStatus` that can be one of *Success*, *Fail*, *Timeout* and *Error*. The *AggregatedTestCaseResult* class adds up to the *StatusTestCaseResult* class by providing an aggregated view on the individual errors of a test case with the properties `:resultCount` and `:resultPrevalence`.

For the individual error reporting the *RLOGTestCaseResult* generates logging messages through the RLog ontology. For every violation, we report the erroneous resource (`rlog:resource`), a message (`rlog:message`) and a logging level (`rlog:level`). The logging level is retrieved from the *TestCase*.

The *ExtendedTestCaseResult* class extends *RLOGTestCaseResult* by providing additional properties for error debugging by reusing the spin ontology. In detail, an *ExtendedTestCaseResult* is a subclass of `spin:ConstraintViolation` and may have the following properties:

- `spin:violationRoot`: the erroneous resource.
- `spin:violationPath`: the property of the resource that the error occurs.
- `:errorPropertyContext`: lists additional properties that may provide a better context for fixing the error. For example, in the `dbo:birthDate` before a `dbo:deathDate` case, `dbo:birthDate` can be the `spin:violationPath` and `dbo:deathDate` the `:errorPropertyContext`.
- `:errorClassification`: is a sub-property of `dct:subject` that points to a SKOS error classification category.
- `:errorSource`: is a sub-property of `dct:subject` that points to a SKOS error source category. Example values can be data parsing, data publishing, mapping, pre processing, post processing, etc.
- `:errorType`: is a sub-property of `dct:subject` and that points to a SKOS error type category on the triple level. Example values can be: missing property, redundant property, inaccurate property.

The extended error annotation is generated through the *ResultAnnotation* class that is attached to a *TestCase* through the `:resultAnnotation` property. A ResultAnnotation must contain an `:annotationProperty` linking to one of the allowed *ExtendedTestCaseResult* properties and an appropriate value for `:annotationValue`. For the schema-based automatic test case generation some of the annotation may be known only on the *Pattern* level and other on the *TestAutoGenerator* level. Thus, *Re-*

sultAnnotations are allowed in both classes and the error annotation are added up on the test case generation.

Finally, we provide `:testSuite`, an ontology annotation property, that links an ontology to an appropriate *TestSuite* for data validation purposes.

```
<http://example.com/ontology#>
    a owl:Ontology ;
    tddo:testCase <http://example.com/testCase> .
```

PATTERN ELICITATION AND CREATION

To start capturing patterns of real data errors we had a closer look at DBpedia, being one of the bigger and best interlinked datasets in the LOD cloud (Lehmann et al., 2015). We performed three different analyses which led to a comprehensive library of test case patterns summarized in Table 19:

1. Analysis of incidental error reports by the DBpedia user community.
2. Analysis of error tracking behavior by Wikipedia editors.
3. Analysis of the ontology schema of the DBpedia OWL ontology.

Community feedback. We thoroughly reviewed all the DBpedia related mailing lists and QA websites, i.e. the *DBpedia discussion*[5] and *DBpedia developers*[6] lists, as well as questions tagged with *DBpedia* on *stackoverflow*[7] and *Semantic Web Answers*[8]. We picked all the data quality related questions and tried to create SPARQL queries for retrieving the same erroneous data. Finally, we grouped similar SPARQL queries together.

Wikipedia maintenance system. We reviewed the information Wikipedia uses to ensure article quality and tried to reuse it from DBpedia. Such information encompasses special *Categories* and *Templates* used by seasoned Wikipedians (e.g. admins and stewards) to administrate and tag errors in the article space[9]. Based on the maintenance categories and templates used, new patterns like the TRIPLE Pattern and the PVT Pattern were derived. These patterns are also applicable to other datasets, e.g. LinkedGeoData (Stadler et al., 2012).

OWL ontology analysis. The main purpose of OWL is to infer knowledge from existing schemata and data. While it can also be used to check constraints, this can be difficult in practice due to the

5 https://lists.sourceforge.net/lists/listinfo/dbpedia-discussion
6 https://lists.sourceforge.net/lists/listinfo/dbpedia-developers
7 http://stackoverflow.com/questions/tagged/dbpedia
8 http://answers.semanticweb.com/tags/dbpedia/
9 http://en.wikipedia.org/wiki/Category:Wikipedia_maintenance

Open World Assumption used and the lack of the Unique Name Assumption. Therefore, in addition to standard OWL inference, it can also be useful to convert OWL ontology axioms to SPARQL queries, which check the constraints expressed by them. This is motivated by research on the *Pellet Integrity Constraint Validator*[10] using the same idea. Specifically, we analysed the ontology and checked which existing constructs are applicable for constraint checking in DBpedia. We identified constructs such as (inverse) functionality, cardinality, domain and range of properties as well as class disjointness as relevant and included them in our pattern template library. The bindings for those patterns can be created automatically from specific OWL ontology axioms.

Pattern Library

Our *Pattern Library* consists of 17 DQTPs. Table 19 shows a description of all patterns along with two example bindings.

Pattern	Description	Type	Binding example
COMP	Comparison between two literal values of a resource.	dom ran pdep	a) `dbo:deathDate` before `dbo:birthDate`
			b) `dbo:releaseDate` after `dbo:latestReleaseDate`
MATCH	The literal value of a resource matches/ does not match a certain regex pattern	ran	a) `dbo:isbn` does not match "^[0-9]{5}$"
			b) `foaf:phone` contains any letters ("[A-Za-z]")
LITRAN	The literal value of a specifically typed resource must (not) be within a given range	ran pdep dom mem	a) `dbo:height` of a `dbo:Person` is not within [0.4,2.5]
			b) `geo:lat` of a `spatial:Feature` is not within [-90,90]
TYPE-DEP	Type dependency: The type of a resource may imply the attribution of another type.	dom cdep	a) a resource is a `gml:_Feature` but not a `dbo:Place`
			b) a resource is a `foaf:Person` but not a `dbo:Person`
TYPRO-DEP	A resource of a specific type should have a certain property.	dom mem pdep	a) a `foaf:Document` should have a `foaf:primaryTopic`
			b) a `dbo:Person` should have a `dbo:birthDate`
PVT	If a resource has a certain value V assigned via a property P_1 that in some way classifies this resource, the existence of another property P_2 can be assumed.	dom pdep	a) DBpedia articles stemming from a *Geographic_location* template must have coordinates assigned via `georss:point`
			b) DBpedia resources in the category *1907 births* should have a `dbo:birthDate`

10 http://clarkparsia.com/pellet/icv/

TRIPLE	A resource can be considered erroneous if there are corresponding hints contained in the dataset		a) resources stemming from maybe copy-pasted Wikipedia articles having the category *Possible_cut-and-paste_moves* b) geographical features (of the linkedgeodata.org dataset) that are marked with the `lgdo:fixme` property
ONE-LANG	A literal value should contain at most one literal for a certain language.	ran card	a) a resource should only have one English `foaf:name` b) a resource should only have one English `rdfs:label`
RDFS-DOMAIN	The attribution of a resource's property (with a certain value) is only valid if the resource is of a certain type.	dom pdep mem	a) a resource having a `dbo:demographicsAsOf` property not being a `dbo:PopulatedPlace` b) a resource has the "Cities of Africa" category assigned but is not of type `dbo:City`
RDFS-RANGE	The attribution of a resource's property is only valid if the value is of a certain type	ran pdep mem	a) a `dbo:Person`'s spouse not being a `dbo:Person` b) a resource assigned via the `foaf:based_near` property not being of type `geo:SpatialThing`
RDFS-RANGED	The attribution of a resource's property is only valid if the literal value has a certain datatype	ran pdep mem	a) the value of the property `dbo:isPeerReviewed` must be of type `xsd:boolean` b) the value of the property `dbo:successfulLaunches` must be of type `xsd:nonNegativeInteger`
INV-FUNC	Some values assigned to a resource are considered to be unique for this particular resource and must not occur in connection with other resources.	ran	a) there must not be more than one resource with the same `foaf:homepage` b) there must not be more than one country with the same `dbo:capital`
OWL-CARD	Cardinality restriction on a property	ran card	a) `dbo:birthDate` is a functional property b) there should be just one `skos:prefLabel`
OWL-DISJC	Disjoint class constraint	cdep	a) a `foaf:Document` is disjoint with `foaf:Project` b) a `dbo:Person` is disjoint with `dbo:Work`
OWL-DISJP	Disjoint property constraint	dom ran pdep mem	a) `skos:prefLabel` is disjoint with `skos:hiddenLabel` b) `dbo:bandMember` is disjoint with `dbo:birthPlace`
OWL-ASYMP	Asymmetric property constraint	dom ran	a) `dbo:child` is asymmetric

			b) `dbo:birthPlace` is asymmetric
OWL-IRREFL	Irreflexive property constraint	dom	a) `dbo:parent` is irreflexive
		ran	b) `dbo:child` is irreflexive

Table 19: Example templates and bindings. The column *Type* refers to the coverage type.

COMP PATTERN Depending on the property semantics, there are cases where two different literal values must have a specific ordering with respect to an operator. *P1* and *P2* are the datatype properties we need to compare and *OP* is the comparison operator R(*OP*) = { `<`, `<=`, `>`, `>=`, `=`, `!=` }.

```
SELECT ?s WHERE { ?s  %%P1%%  ?v1 .
                  ?s  %%P2%%  ?v2 .
                  FILTER ( ?v1 %%OP%% ?v2 ) }
```

Example bindings:

1. `dbo:deathDate` before '<' `dbo:birthDate`
2. `dbo:releaseDate` after '>' `dbo:latestReleaseDate`
3. `dbo:demolitionDate` before '<' `dbo:buildingStartDate`

MATCH PATTERN Application logic or real world constraints may put restrictions on the form of a literal value. *P1* is the property we need to check against *REGEX* and *NOP* can be a not operator ('!') or empty.

```
SELECT DISTINCT ?s WHERE { ?s  %%P1%%  ?value .
   FILTER ( %%NOP%% regex(str(?value), %%REGEX%) ) }
```

Example bindings:

1. `dbo:isbn` format is different '!' from "^([iIsSbBnN 0-9-])*$"
2. `dbo:postCode` format is different '!' from "^[0-9]{5}$"
3. `foaf:phone` contains any letters ("[A-Za-z]")

LITRAN PATTERN Application logic or real world facts may put restrictions on the range of a literal value depending on the type of a resource. *P1* is a property of an instance of class *T1* and its literal value must be between the range of [*Vmin*,*Vmax*] or outside (*NOP* can be a '!'). The query is phrased so that "between" does not require negation, but "outside" does.

```
SELECT DISTINCT ?s WHERE {
            ?s  rdf:type  %%T1%% .
            ?s  %%P1%%    ?value .
            FILTER( %%NOP%%
                    (?value < %%Vmin%% ||
                     ?value > %%Vmax%%))) }
```

Example bindings:

1. a `dbo:Person` should have `dbo:height` between 0.4 and 2.5 meters
2. the `geo:lat` of a `gml:_Feature` must be in range [-90,90]
3. the `geo:long` of a `gml:_Feature` must be in range [-180,180]

TYPEDEP PATTERN The type of a resource may imply the attribution of a second type. In this pattern *T1* and *T2* are the types tested for coexistence.

```
SELECT DISTINCT ?s WHERE {
        ?s  rdf:type  %%T1%% .
        FILTER NOT EXISTS { ?s rdf:type  %%T2%% } }
```

Example bindings:

1. `gml:_Feature` should imply `dbo:Place`
2. `yago:GeoclassCapitalOfAPoliticalEntity` should imply `dbo:Place`
3. `foaf:Person` should imply `dbo:Person`

TYPRODEP PATTERN Resources of a given type sometimes must be accompanied by a specified property. In this pattern the type *T1* is tested for coexistence with property *P1*.

```
SELECT DISTINCT ?s WHERE {
        ?s  rdf:type  %%T1%% .
        FILTER NOT EXISTS { ?s  %%P1%%  ?v } }
```

Example bindings: Resources representing

1. a `dbo:Place` should have a `geo:lat` property
2. a `dbo:Person` should have a `dbo:birthDate` property
3. a `dbo:Person` should have a `foaf:depiction` property

PVT PATTERN If a resource has a certain value *V* assigned via a property *P1* that in some way classifies this resource, one can assume the existence of other properties *P2*. The following pattern provides the test template for such cases.

```
SELECT DISTINCT ?s WHERE {
        ?s  %%P1%%  %%V1%%
        FILTER NOT EXISTS { ?s  %%P2%%  ?p } }
```

Example bindings: Resources

1. being extracted from a `dpt:Template:Geographic_location` should have a geo coordinate assigned (`dbo:georss:point`)

2. belonging to the category `dbc:1907_births` should have a `dbo:birthDate`
3. belonging to a Wikipedia category for maintenance, because they are using a template (`dbp:wikiPageUsesTemplate dbt:Infobox_character`), but have unlabeled fields (i.e. missing properties such as `dbpprop:first`)[11]

TRIPLE PATTERN In some cases hints with regards to errors or bad smells are already contained in the dataset. These are given as certain property *P1* value *V1* combinations and can be tested with the following pattern.

```
SELECT DISTINCT ?s WHERE { ?s  %%P1%%  %%V1%% }
```

Example bindings: Resources extracted from Wikipedia articles, that

1. were possibly copy-pasted (`dc:subject dbc:Possible_cut-and-paste_moves`)
2. have an inconsistent citation format (`dbp:wikiPageUsesTemplate dbt:Inconsistent_citations`)
3. have missing files (`dc:subject dbc:Articles_with_missing_files`)

ONELANG PATTERN A literal value should contain at most 1 literal for a language. *P1* is the property containing the literal and *V1* is the language we want to check.

```
SELECT DISTINCT ?s WHERE { ?s  %%P1%%  ?c
       BIND ( lang(?c) AS ?l )
       FILTER (isLiteral (?c) && lang(?c) = %%V1%%)}
GROUP BY ?s HAVING COUNT (?l) > 1
```

Example bindings:

1. a single English ("en") `foaf:name`
2. a single English ("en") `rdfs:label`

RDFSDOMAIN PATTERN The attribution of a property is only valid when the class is in the domain of the property. In this pattern the property *P1* is tested for coexistence of the type *T1*. Optionally value *V1* can be specified to narrow the test to the specified value for *P1*.

```
SELECT DISTINCT ?s WHERE { ?s  %%P1%%  %%V1%% .
       FILTER NOT EXISTS {?s rdf:type ?T1 .
         ?T1 rdfs:subClassOf%%OP%% %%T1%% . }
       FILTER NOT EXISTS {?s rdf:type %%T1%%  } }
```

Example bindings:

1. `dc:subject dbc:CapitalsInAfrica` should have type `dbo:Place` attributed
2. `dbo:dissolved` should have type `dbo:SoccerClub` attributed

11 http://en.wikipedia.org/wiki/Category:Articles_using_Infobox_character_with_multiple_unlabeled_fields

RDFSRANGE PATTERN The object of a triple must be within the range of the property. In this pattern property *P1* and type *T1* are tested for coexistence.

```
SELECT DISTINCT ?s WHERE { ?s  %%P1%%  ?c .
       FILTER NOT EXISTS {?c rdf:type ?T1 .
         ?T1 rdfs:subClassOf%%OP%% %%T1%% . }
       FILTER NOT EXISTS {?c rdf:type %%T1%%  } }
```

Example bindings:

1. the `dbo:spouse` of a `dbo:Person` must be a `dbo:Person`
2. the `dbo:birthPlace` of a `dbo:Person` must be a `dbo:Place`
3. the `dbo:dean` of a `dbo:EducationalInstitution` must be a `dbo:Person`

RDFSRANGED PATTERN The (literal) object of a triple must be of a certain datatype determined by the property used. In this pattern the property *P1* and the datatype *D1* are tested for coexistence.

```
SELECT DISTINCT ?s WHERE {
      ?s %%P1%% ?c.
      FILTER (DATATYPE(?c) != %%D1%%) }
```

Example bindings:

1. the value of the property `dbo:certificationDate` must be of type `xsd:date`
2. the value of the property `dbo:isPeerReviewed` must be of type `xsd:boolean`
3. the value of the property `dbo:successfulLaunches` must be of type `xsd:nonNegativeInteger`

INVFUNC PATTERN Some values assigned to a resource are considered to be unique for this particular resource and should not occur in connection with other resources. This pattern can be extended to also restrict the value as shown in the comments of the following listing.

```
SELECT DISTINCT ?s WHERE{
       ?a  %%P1%%  ?v1 . # ?a %%P2%%  %%V1%% .
       ?b  %%P1%%  ?v2 . # ?b %%P2%%  %%V1%% .
       FILTER ((str(?v1) == str(?v2)) && (?a != ?b))}
```

Example bindings:

1. two different resources should not have the same `foaf:homepage` (*P1*, *P2*)
2. two countries should not have the same `dbo:capital`

OWLCARD PATTERN Using this pattern, we can test for cardinal constraints on specific properties. *P1* is the property we need to compare with *V1* and *OP* is the comparison operator (`<`, `<=`, `>`, `>=`, `=`, `!=`)

```
SELECT DISTINCT ?s WHERE { ?s %%P1%% ?c }
GROUP BY ?s HAVING count(?c) %%OP%% %%V1%%
```

Example bindings:

1. every property defined as `owl:FunctionalProperty` (e.g. `dbo:birthDate`, `dbo:latestReleaseDate`) in the ontology cannot exist more than once (>1)
2. dbpedia.org's resources have an `rdfs:label` for each of its 20 different languages. Therefore each resource should not have more than 20 labels (>20), the same holds for other properties such as `rdfs:comment`.

OWLDISJC PATTERN A resource must not belong to two disjoint classes. *T1* and *T2* are the two disjoint classes we check.

```
SELECT DISTINCT ?s WHERE {
            ?s  rdf:type  %%T1%% .
            ?s  rdf:type  %%T2%% . }
```

Example bindings: (a) `dbo:Person` is `owl:disjointWith` with `dbo:Place`, (b) `dbo:Person` is `owl:disjointWith` with `dbo:Work`,

OWLDISJP PATTERN A triple object *v* cannot be assigned to a resource *s* via both properties *P1* and *P2* if these are stated to be disjoint by an `owl:disjointProperty` axiom.

```
SELECT DISTINCT ?s WHERE { ?s %%P1%% ?v .
                           ?s %%P2%% ?v .}
```

Example bindings:

1. `skos:prefLabel` is disjoint with `skos:hiddenLabel`
2. `dbo:bandMember` is disjoint with `dbo:birthPlace`

OWLASYMP PATTERN For a given property *P1* that is declared to be asymmetric, this pattern checks if there are violating cases where it is nonetheless used as symmetric property, i.e. for two resources *a* and *b* there are axioms for *a P1 b.* and *b P1 a.* .

```
SELECT ?r1 WHERE { ?r1 %%P1%% ?r2 .
                   ?r2 %%P1%% ?r1 . }
```

Example bindings:

1. child parent relations (`dbo:child`) cannot be symmetric
2. person birth place relations (`dbo:birthPlace`) cannot be symmetric

OWLIRREFL PATTERN For a given property *P1* that is declared to be irreflexive, this pattern find violating statements that nonetheless use this property reflexively, i.e. for a resource *a* there is an axiom *a P1 a.* .

```
1 SELECT DISTINCT ?s WHERE {?s %%P1%% ?s .}
```

Example bindings:

1. a resource cannot be its own parent (`dbo:parent`)
2. a resource cannot be its own child (`dbo:child`)

CONCLUSIONS AND FUTURE WORK

In this chapter, we described a novel approach for assessing and improving Linked Data quality. The approach is inspired by test-driven software engineering and is centred around the definition of data quality integrity constraints, which are represented in SPARQL query templates.

We see this work as the first step in a larger research and development agenda to position test-driven data engineering similar to test-driven software engineering. As a result, we hope that test-driven data quality can contribute to solve one of the most pressing problems of the Data Web – the improvement of data quality and the increase of Linked Data fitness for use.

9

TEST-DRIVEN QUALITY ASSESSMENT EVALUATION

Kontokostas, Westphal, Auer, Hellmann, Lehmann, and Cornelissen, (2014a,b,c)

In this chapter we implement and evaluate the *Test-Driven Quality Assessment* methodology that was described in the previous chapter (Chapter 8).

Section 9.1 describes the implementation of the methodology in a software tool called RDFUnit. We provide technical and architectural details as well as ways the tool can be invoked.

In Section 9.2 we use RDFUnit to perform automatic test case instantiations for all available schemata registered with the *Linked Open Vocabularies* (LOV)[1]. This resulted in 32,293 total unique and reusable test cases for 297 of the LOV vocabularies, independent of their domain or their purpose. .

A core contribution of this work is the extensive and unprecedented quantitative evaluation involving manual and automatic test case instantiations for five large-scale LOD datasets (two DBpedia editions, datos.bne.es, Library of Congress authority data and LinkedGeoData) in Section 9.3.

Additionally, we show progress in implementing domain-specific validation by quickly improving existing validation provided by ontology maintainers. We specifically analysed datasets for two emerging domain ontologies, the *lemon model* (McCrae et al., 2012) and the *NIF 2.0 Core Ontology* (Hellmann et al., 2013a) and evaluated 11 datasets in Section 9.4.3 using automatic and manual test cases. Using the ontology, we annotate test cases and provide support for different levels of result reporting allowing to give feedback to developers when running these tests and ultimately improving data quality.

One of the main advantages of our approach is that domain specific semantics can be encoded in the data quality test cases, thus being able to discover data quality problems beyond conventional quality heuristics. Finally, our framework implementation is built upon the SPARQL 1.1 standard which makes it applicable for any knowledge bases or triple store implementation. We conclude in Section 9.5.

IMPLEMENTATION, ARCHITECTURE AND EXTENSIBILITY OF RDFUNIT

Kontokostas, Westphal, Auer, Hellmann, Lehmann, and Cornelissen, (2014a)

RDFUnit (formerly known as Databugger) is a tool built to showcase the *test-driven quality assessment methodology*. The tool is released as open source under the Apache License and provides both, a com-

1 http://lov.okfn.org/

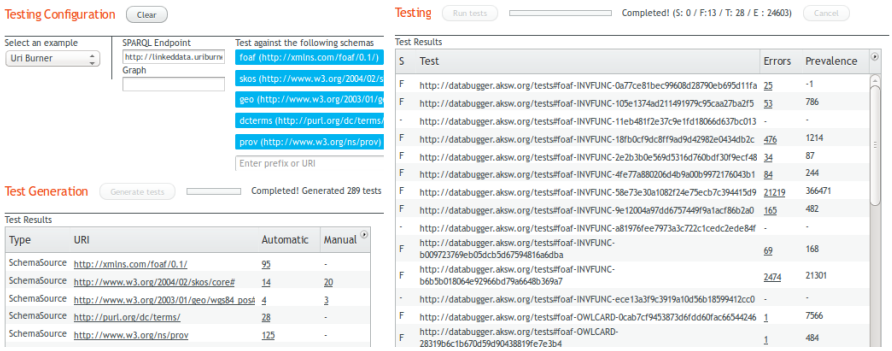


Figure 25: Screenshot of the RDFUnit web interface. In the upper left section the user configures the SPARQL Endpoint, the graph and the schemas she wants to test her data with. In the lower left section, test cases are automatically generated by parsing the schemas. Manual tests predefined for a schema are also loaded. In the right section RDFUnit starts running the tests and displays test results to the user.

mand line interface (CLI) and a web interface (cf. Figure 25)[2,3]. A simple CLI test configuration can be generated with:

```
$ rdfunit -d <dataset-uri> -e <endpoint-uri>
          [-g <graph1|graph2|...>  ]
         -s <schema-prefix1,schema-prefix2,...>
```

Once the user starts a test configuration, the framework dereferences and reads the schemas. For all the provided schemas, RDFUnit generates automatic test cases using TAGs and loads any existing manual test cases for the schemas or the dataset. All the test cases are then executed against the SPARQL endpoint. The test results are both displayed on the screen and stored in RDF. The web interface allows the user to generate the same test configuration from a more interactive interface (cf. Figure 25).

The DQTPs[4], the manual and auto-generated test cases[5], the TAGs[6] and the test results are modeled under the RDFUnit ontology (cf. Section 8.2. The input and output of the tool is entirely in RDF which makes it highly configurable. One can read the input (patterns, TAGs and test cases) from the file system as RDF files, dereference then from a remote location or retrieve them from a SPARQL endpoint. Concrete test cases are equipped with persistent identifiers to facilitate test tracking over time. Our pattern library uses SPARQL1.1 and *property paths*[7] for properly checking transitive violations (e.g. `rdfs:domain` and `rdfs:range`).

2 http://rdfunit.aksw.org
3 A screencast of the tool is available at http://youtu.be/3g9R3P1kwdw
4 http://rdfunit.aksw.org/data/patterns#
5 http://rdfunit.aksw.org/data/tests#
6 http://rdfunit.aksw.org/data/generators#
7 http://www.w3.org/TR/sparql11-property-paths/

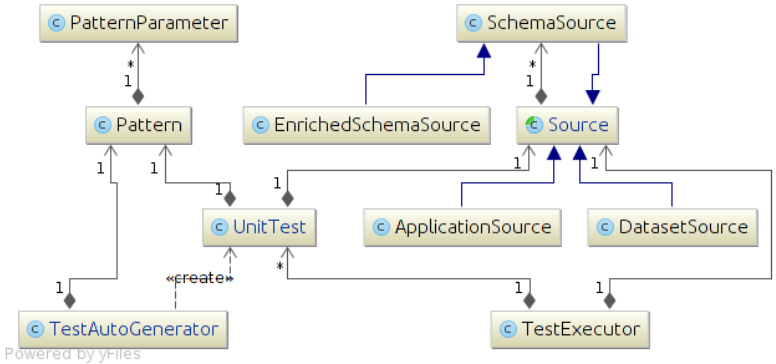


Figure 26: The main components of the core RDFUnit library as a UML diagram. The diagram was generated with the Intellij IDEA program.

RDFUnit was built with *Java* and the *Jena* framework. A UML diagram of the core library is depicted in Figure 26. The main components of the RDFUnit Library are the *Source*, *Pattern*, *TestAutoGenerator* and *UnitTest*. A *Source* represents an arbitrary RDF source uniquely identified by a URI. Concrete *Source* implementations are:

- *SchemaSource*: a schema, a vocabulary or an ontology in RDF, e.g. `skos`. The framework can automatically dereference a schema from a URI, a file location or a prefix (e.g. `foaf`). For prefix resolution we query the LOV SPARQL endpoint[8] to get a dereferenceable URI. Using LOV we provide easy test access to commonly used vocabularies.
- *EnrichedSchemaSource*: a semi-automatically enriched schema.
- *DatasetSource*: an RDF dataset accessible via a SPARQL endpoint, e.g. `DBpedia`[9]. A subset of the dataset can be selected by providing a list of Named Graphs. RDF dump datasets will be supported in the following releases of RDFUnit.

The *Pattern* component holds a DQTP and *UnitTests* are instantiations of *Patterns* generated by binding a pattern placeholder to valid replacements. According to our methodology, a *UnitTest* can be created either manually or automatically. The automatic *UnitTest* generation is performed by the *TestAutoGenerator* (TAG) component. Each TAG is based on a *Pattern* and automatically instantiates test cases (*UnitTests*) for an input schema.

The *TestGeneratorExecutor* component takes as input a dataset and a list of schemas, for each schema a) automatically generating test cases and b) loading any existing manually defined test cases. Additionally,

8 http://lov.okfn.org
9 http://dbpedia.org

Schema	Test Cases	Schema	Test Cases
dicom	8,229	mo	605
dbo	5,713	tio	525
frbrer	2,166	uco	516
biopax	688	vvo	506
hdo	682	ceo	511

Table 20: Top 10 schemas with descending number of automatically generated test cases.

any pre-defined test cases for the dataset are loaded. The automatically generated test cases are cached locally for future reference. The output of this component is passed to the *TestExecutor* component which executes the test cases against a *DatasetSource*. The *TestCoverageEvaluator* is an optional step that calculates the test coverage of a test case set against a dataset. However, this step requires precalculated property and class statistics of the dataset. Future versions of the library will be able to autogenerate these statistics.

TEST GENERATION

To evaluate our methodology, we automatically generated test cases for all available vocabularies in the LOV dataset. Using the implemented TAGs, we managed to create 32,293 total unique reusable test cases for 297 LOV vocabularies[10]. Table 20 displays the 10 schemas with the most associated test cases. For brevity we use the vocabulary prefixes as defined in LOV[11]. Test cases are themselves described in RDF and have a stable URI for tracking them over time. The URI is generated under the application namespace concatenated with the schema prefix, the pattern and an MD5 checksum of the SPARQL query string. The following listing displays a test case that checks whether the `rdfs:range` of `foaf:isPrimaryTopicOf` instance is a `foaf:Document`. We store metadata along with every test case which allows us to easily filter test cases based on different criteria.

```
tddt:foaf-RDFSDOMAIN-8e121cf1111201b5d53de161e245c13
a tddo:PatternBasedTestCase ;
tddo:appliesTo tddo:Schema ;
tddo:generated tddo:AutoGenerated ;
tddo:source <http://xmlns.com/foaf/0.1/> ;
tddo:references foaf:Document,foaf:isPrimaryTopicOf;
tddo:testGenerator tddg:RDFSRANGEC .
tddo:basedOnPattern tddp:RDFSRANGE ;
tddo:binding [ ... ] ;
```

10 LOV had 367 vocabularies at the date of last access (5/10/2013) but not all were accessible.

11 In addition to the LOV schemas, dbo (http://dbpedia.org/ontology/), frbrer (http://iflastandards.info/ns/fr/frbr/frbrer/) and isbd (http://iflastandards.info/ns/isbd/elements/) schemas are included as prefixes.

Schema	TC	Schema	TC
dbpedia.org	1,723	id.loc.gov	48
nl.dbpedia.org	845	datos.bne.org	18
linkedgeodata.org	61		

Table 21: Number of additional test cases (TC) instantiated for the enriched schemas.

```
10 tddo:testCaseLogLevel rlog:Error .
```

For every dataset evaluated, we applied automatic schema enrichment as described in Section 8.1. We used a high level of confidence (0.9; see (Bühmann and Lehmann, 2013) for details) on the produced axioms and applied manual post-processing to remove certain axioms. The number of additional test cases instantiated for the considered schemas are shown in Table 21.

Besides the automatically generated test cases, our methodology supports manual test cases that may apply to a schema or a dataset. The manual schema test cases are reusable across different datasets for all RDF data using that schema. The manual dataset test cases can be applied only to a specific dataset. Manual test cases usually require domain knowledge, which the authors have for a subset of the evaluation datasets. For the purposes of this evaluation, we defined 22 manual test cases for the DBpedia ontology (`dbo`), six for the LinkedGeoData ontology (`lgdo`), three for the WGS84 Geo Positioning ontology (`geo`) and 15 manual test cases for the DBpedia in English dataset. Additionally, we defined 20 manual test cases for the `SKOS` vocabulary exploiting existing domain expertise (Suominen and Hyvönen, 2012). Table 22 presents an aggregation of the defined test cases based on the pattern they stem from.

SEMI-AUTOMATED, LARGE-SCALE LINKED DATA QUALITY EVALUATION

Kontokostas, Westphal, Auer, Hellmann, Lehmann, Cornelissen, and Zaveri, (2014c)

To showcase the re-usability of our automatically and manually generated test cases, we chose the following datasets for evaluation:

- `dbpedia.org`[12] extracts data from the English Wikipedia and publishes the data using the following schemas: `owl`, `dbo`, `foaf`, `dcterms`, `dc`, `skos`, `geo` and `prov` (Lehmann et al., 2015).
- `nl.dbpedia.org`[13] extracts data from the Dutch Wikipedia edition using the same vocabularies as the English DBpedia.

12 {http://dbpedia.org}(version3.9)
13 {http://nl.dbpedia.org}(liveversion,accessedon05/10)

Pattern	Test Cases	Manual
RDFSDOMAIN	16,645	3
RDFSRANGE	9.727	4
OWLDISJC	5,530	-
EDFSRANGED	5,073	-
OWLDISJP	1,813	10
OWLCARD	1,818	6
TYPRODEP	746	13
OWLASYMP	660	-
OWLIRREFL	342	-
INVFUNC	338	1
MATCH	9	9
LITRAN	5	5
COMP	4	4
ONELANG	4	4
PROPDEP	4	4
TYPDEP	2	2
TRIPLE	2	2

Table 22: Number of total and manual test cases per pattern for all LOV vocabularies.

- `linkedgeodata.org`[14] provides a linked data mirror of OpenStreetMap[15] using the following schemas: `ngeo`, `spatial`, `lgdo`, `dcterms`, `gsp`, `owl`, `geo`, `skos` and `foaf` (Stadler et al., 2012).
- `id.loc.gov`[16] is a SKOS dataset that publishes Library of Congress authority data using `owl`, `foaf`, `dcterms`, `skos`, `mads`, `mrel` and `premis` schemas.
- `datos.bne.es`[17] provides open bibliographic linked data from the Spanish National Library using `owl`, `frbrer`, `isbd`, `dcterms` and `skos` schemas.

To identify the schemas for each dataset, we used existing information from the *LODStats* project[18] (Demter et al., 2012). The English (`dben`) and Dutch (`dbnl`) DBpedia share a similar structure, but the actual data differs (Kontokostas et al., 2012). Both DBpedia and the LinkedGeoData (`lgd`) datasets are generated from crowdsourced content and thus are prone to errors. The Library of Congress authority data (`loc`) and the Open bibliographic data from the Spanish National Library (`datos`) were chosen as high quality bibliographic datasets with `loc` focusing on publishing SKOS and in the case of

14 http://downloads.linkedgeodata.org/releases/2013-08-14/
15 http://www.openstreetmap.org
16 {http://id.loc.gov/download/}(accessedon05/10/2013)
17 {http://datos.bne.es/datadumps/},(accessedon05/10/2013)
18 http://stats.lod2.eu/

Dataset	Triples	Subjects	TC	Pass	Fail	TO	Errors	ManEr	EnrEr	E/R
DBp-en	817.5M	24.9M	6K	4.2K	1.9K	55	63.6M	5.2M	249.8K	2.55
DBp-nl	74.8M	4.8M	5.2K	4.1K	812	73	5.4M	211.6K	15K	1.11
LGD	274.7M	51.9M	634	545	86	3	57.7M	133.1K	1	1.11
Datos	60M	7.4M	2.5K	2.4K	89	8	27.9M	25	537	3.74
LoC	436.1M	53.1M	536	499	28	9	9.4M	49	3.7K	0.18

Table 23: Evaluation overview for the five tested datasets. For every dataset we display the total number of triples and the distinct number of subjects. We mention the total number of test cases (TC) that were run on each dataset, how many tests passed, failed and timed out (TO). Finally we show the total number of errors, as well the total number of errors that occurred from manual (ManEr) and enriched (EnrEr) tests. The last column shows the average errors per distinct subject.

`datos` FRBR[19] data. The DBpedia datasets were tested using their online SPARQL endpoints and the other three datasets were loaded in a local triple store[20].

Table 23 provides an overview of the dataset quality evaluation. In Table 24 we present the total errors aggregated per schema and in Table 25 the total errors aggregated per pattern. The test coverage for every dataset is provided in Table 26.

The occurrence of a high number of errors in the English DBpedia is attributed to the data loaded from external sources. For example, the recent load of transformed Wikidata data[21] almost doubled the `rdfs:domain` and `rdfs:range` violations and errors in the `geo` schema. A common error in DBpedia is the `rdfs:range` violation. Triples are extracted from data streams and complete object range validation cannot occur at the time of extraction. Example violations from the `dbo` schema are the `rdfs:domain` of `dbo:sex` (1M) and `dbo:years` (550K) properties. Other `dben` errors based on the `foaf` schema are attributed mainly to the incorrect `rdfs:domain` or `rdfs:range` of `foaf:primaryTopic` (12M), `foaf:isPrimaryTopicOf` (12M) `foaf:thumbnail` (3M) and `foaf:homepage` (0.5M).

Among errors from the manual test cases created for the DBpedia ontology are the following:

- 163K (102K in `dbnl`) resources with wrong postal code format.
- 7K (137 in `dbnl`) books with wrong ISBN format.
- 40K (1.2K in `dbnl`) persons with a death date and without birth date.
- 638K persons without a birth date.

19 www.oclc.org/research/activities/frbr.html

20 We used the Virtuoso V7 triple store, because it supports SPARQL 1.1 property paths.

21 http://www.mail-archive.com/dbpedia-discussion@lists.sourceforge.net/msg05583.html

- 197K places without coordinates.
- 242K resources with coordinates that are not a `dbo:Place`.
- 28K resources with exactly the same coordinates with another resource.
- 9 resources with invalid longitude.

The `lgd` dataset also has a high number of errors per resource. Although the LinkedGeoData ontology is big, the information of interest for our methodology is mostly limited to `rdfs:domain` and `rdfs:range` axioms. Due to its broad vocabulary, mostly stemming from curated crowdsourced user input, only a few manual test cases were found. In-depth domain knowledge is required to define further test cases. These resulted in 132K errors for resources with a `lgdo:fixme` predicate and 250 with `lgdo:todo`, 637 wrong phone numbers and 22 resources having a `lgdo:start` property but no `lgdo:end`.

The `datos` dataset yielded a total of 28 million errors. In absolute numbers, `rdfs:domain` and `rdfs:range` violations were dominant. The `isbd:P1016` and `isbd:P1185` properties produced the most `rdfs:domain` violations (2.38M and 2.35M respectively). The schemas used in `datos` are expressive and there were many violations stemming from `owl:disjointWith` and `owl:propertyDisjointWith` constraints. With regards to the manual errors, 6 occurred due to shared literals between `skos:prefLabel` and `skos:altLabel` (Suominen and Hyvönen, 2012) and 25 because of property disjointness violations between `skos:broader`, `skos:narrower` and `skos:related`.

The `loc` dataset generated a total of 9 million errors. However, 99.9% originated from one test case: the `rdfs:domain` of `skos:member`. Other minor errors occurred in other schemas (cf. Table 24), e.g. incorrect `rdfs:domain` of `skos:topConceptOf` and incorrect `rdfs:domain` of `foaf:focus`. Similar to the `datos` dataset, 49 manual errors occurred from disjoint properties between `skos:broader`, `skos:narrower` and `skos:related`.

The highest *test coverage* is found in the `datos` dataset. This is due to the rich `frbrer` and `isbd` schemas. Although `dben` had a bigger test set than `datos`, it publishes a lot of automatically generated properties under the `dbp` namespace Lehmann et al., 2015, Section 2 which lowers the coverage scores. The low test coverage for `lgd` can be attributed to the very large but relatively flat and inexpressive schema. For DBpedia in Dutch we evaluated the Live endpoint and thus could not calculate property and class occurrences.

Discussion

The most frequent errors in all datasets were produced from `rdfs:domain` and `rdfs:range` test cases. Domain and range are two of the most

		Errors				
Schema	**TC**	**dben**	**dbnl**	**lgd**	**dat.**	**loc**
dbo	5.7K	7.9M	716K	-	-	-
frbrer	2.1K	-	-	-	11K	-
lgdo	224	-	-	2.8M	-	-
isbd	179	-	-	-	28M	-
prov	125	25M	-	-	-	-
foaf	95	25M	4.6M	-	-	59
gsp	83	-	-	39M	-	-
mads	75	-	-	-	-	0.3M
owl	48	5	3	2	5	-
skos	28	41	-	-	-	9M
dcterms	28	960	881	191K	37K	659
ngeo	18		-	119	-	-
geo	7	2.8M	120K	16M	-	-

Table 24: Total errors in the evaluated datasets per schema.

commonly expressed axioms in most schemas and, thus, produce many automated test cases and good test coverage. Errors from such violations alone cannot classify a dataset as low quality. In DBpedia, a resource is generated for every Wikipedia page and connected with the original article through the `foaf:primaryTopic`, `foaf:isPrimaryTopicOf` and `prov:wasDerivedFrom` predicates. DBpedia neither states that the Wikipedia page is a `foaf:Document` nor that the DBpedia resource a `prov:Entity`, as the FOAF and PROV vocabularies demand. This produced a total of 33 million errors (35% of the total errors) in the English DBpedia. In most cases, fixing such errors is easy and dramatically reduces the error rate of a dataset. However, DBpedia, as well as most LOD datasets, do not load all the schemas they reference in their endpoints. Thus, locating such errors by using only local knowledge is not effective, whereas our pattern library can be used without further overhead.

Testing for external vocabularies. According to our methodology, a dataset is tested against all the schemas it references. Although this approach provides better testing coverage, it can be insufficient when testing against unused data. Properties like `foaf:weblog` that do not exist in neither evaluated dataset, auto-generate three test cases for `rdfs:domain`, `rdfs:range` and `owl:InverseFunctionalProperty`. In the future, the methodology could be refined to intelligently preprocess a dataset and reduce the number of test cases to run.

Revision of manually instantiated patterns. Although our pattern library already covers a wide range of data quality errors, there are cases where the mere instantiation of patterns is not sufficient. Binding COMP-a (cf. Table 19), for example, returns 509 results in the English DBpedia. Some of these results have, however, incomplete

Pattern	dben	dbnl	lgd	datos	loc
COMP	1.7M	7	-	-	-
INVFUNC	279K	13K	-	511	3.5K
LITRAN	9	-	-	-	-
MATCH	171K	103K	637	-	-
OWLASYMP	19K	3K	-	-	-
OWLCARD	610	291	1	1	3
OWLDISJC	92	-	-	8.1K	1.1K
OWLDISJP	3.4K	7K	-	53	223
OWLIRREFL	1.4K	14	-	-	-
PVT	267K	1.2K	22	-	-
RDFSDOMAIN	31M	2.3M	55M	28M	9M
RDFSRANGE	26M	2.5M	191K	320K	111K
RDFSRANGED	760K	286K	2.7M	2	-
TRIPLE	-	-	132K	-	-
TYPDEP	674K	-	-	-	-
TYPRODEP	2M	100K	-	-	-

Table 25: Total errors per pattern.

Metric	dben	lgd	datos	loc
f_{pdom}	20.32%	8.98%	72.26%	20.35%
f_{pran}	23.67%	10.78%	37.64%	28.78%
f_{pdep}	24.93%	13.65%	77.75%	29.78%
f_{card}	23.67%	10.78%	37.63%	28.78%
f_{mem}	73.51%	12.78%	93.57%	58.62%
f_{cdep}	37.55%	0%	93.56%	36.86%
$Cov(QS, D)$	33.94%	9.49%	68.74%	33.86%

Table 26: Test coverage on the evaluated datasets.

dates (i.e. just `xsd:gMonthDay`). Technically, these results are outside of the scope of the binding and the pattern and, therefore, a false positive. This can only be resolved by writing manual test cases or adding another DQTP. In this scenario, the extended test case could be as follows:

```
SELECT COUNT(*) WHERE { ?s dbo:birthDate ?v1 .
                        ?s dbo:deathDate ?v2 .
  FILTER (?v1>?v2 && datatype(?v1)!=xsd:gMonthDay
                  && datatype(?v2)!=xsd:gMonthDay) }
```

While axioms in an OWL ontology are intended to be applicable in a global context, our test-driven methodology also depends on domain knowledge to capture more semantics in data. However, there are cases where data constraints can be very application specific and not universally valid. For instance, due to the vast size of DBpedia, it is unrealistic to expect completeness, e.g. that every `dbo:Person` has

a `foaf:depiction` and a `dbo:birthDate`. However, in the context of an application like "A day like today in history"[22] these properties are mandatory. Thus, a refinement of the methodology could support manual tests cases associated for an application context.

The software used to generate the test cases and produce the evaluation results is available as open source[23]. At the project website[24], we provide a user interface and a dump of all results as RDF .

DOMAIN-SPECIFIC LINKED DATA QUALITY ASSESSMENT FOR LINGUISTIC ONTOLOGIES

Kontokostas, Brümmer, Hellmann, Lehmann, and Ioannidis, (2014b)

Linked Data (LD) comprises of an unprecedented volume of structured data on the Web and is adopted from an increasing number of domains. However, the varying quality of the published data forms a barrier in further adoption, especially for Linked Data consumers. Natural Language Processing (NLP) is – compared to other domains, such as Biology – a late LD adopter with a steep rise of activity in the creation of vocabularies, ontologies and data publishing. A plethora of workshops and conferences such as LDL `http://ldl2014.org/`, WoLE `http://wole2013.eurecom.fr`, LREC `http://lrec2014.lrec-conf.org`, MLODE `http://sabre2012.infai.org/mlode`, NLP&DBpedia `http://nlp-dbpedia2013.blogs.aksw.org/program/`) motivate researchers to adopt Linked Data and RDF/OWL and convert traditional data formats such as XML and relational databases. Although guidelines and best practices for this conversion exist, developers from NLP are often unfamiliar with them, resulting in low quality and inoperable data. In this paper, we address the subsequently arising need for data quality assessment of those NLP datasets.

In this section, we will discuss the employment of RDFUnit for *lemon* and *NIF*, especially with regard to these questions: (1) What is the coverage of the automatically generated tests, what are their limitations. (2) Where is it feasible to use the predefined patterns from the pattern library (Section 8.3)? Are there test cases that are too complex and need manual creation by an expert? (3) Which test cases can not be expressed at all as they are not expressible via SPARQL?

By running the existing RDFUnit Test Auto Generators (TAG) on the *lemon* and *NIF* ontologies we automatically generated 172 test cases for *lemon* and 86 test cases for *NIF* (cf. Table 27). Both ontologies are of similar size: *NIF* contains 19 classes and 46 properties while *lemon* 23 classes and 55 properties. The number of increased test cases in *lemon* results from the higher amount of defined cardinality and disjointness restrictions. The RDFUnit Suite, at the time of writing, does not provide full OWL coverage and thus,

22 `http://el.dbpedia.org/apps/DayLikeToday/`
23 `http://github.com/AKSW/RDFUnit`
24 `http://rdfunit.aksw.org`

	Total	Dom.	Ran.	Datat.	Card.	Disj.	Func.	I. Func.
Lemon	172	40	34	1	29	64	3	1
NIF	86	42	24	4		6	10	

Table 27: Number of automatically generated test cases per ontology. We provide the total number of test cases as well as separated per `rdfs domain` and `range`, literal datatype, `OWL` cardinality (min, max, exact), property & class disjointness, functional and inverse functional constraints.

complex `owl:Restrictions` cannot be handled yet. In the frame of the examined ontologies, RDFUnit did not produce test cases for unions of (`owl:unionOf`) restrictions such as multiple cardinalities for `lemon:LexicalSense` and `lemon:LemonElement` or restrictions with `owl:allValuesFrom`, `owl:someValuesFrom` and `owl:hasSelf` for *NIF*.

Both *NIF* and *lemon* have defined semantic constraints that can not be captured in OWL and are too complex for the above-mentioned pattern library. In particular, *NIF* and *lemon* use natural language text in the `rdfs:comment` properties as well as their documentations and specification documents.

For *lemon*, the maintainers implemented a Python validator[25], which enables us to directly compare our efforts to a software validator. For *NIF* there was an early prototype of RDFUnit that used only manual SPARQL test cases.

Lemon

According to Table 27, test cases for `rdfs:domain` and `rdfs:range` restrictions are the largest group, at 43.8%, followed by tests for disjointness (37.4%) and cardinality restrictions (18.8%). The existing *lemon* validator contains 24 test cases for some structural criteria of the *lemon* ontology. 14 of these tests are natively covered by the existing RDFUnit TAGs. Out of the 10 remaining cases, four where on warning and info level, based on recommendations from the ontology's guidelines. They are thus not explicitly stated in OWL, because they don't constitute logical errors and can not be covered by automatic test generation. Of the six remaining errors, two where expressed via `owl:unionOf` and two could not be expressed by the ontology's author because OWL is not able to express them. Additionally, the *lemon* validator reported undeclared properties under the *lemon* namespace. Although this test case can be expressed in SPARQL, it was not implemented at the time of writing.

The last error case not covered was due to an error in the ontology itself. *Lemon* defines that every instance of `lemon:LexicalEntry`

25 https://github.com/jmccrae/lemon-model.net/blob/master/validator/lemon-validator.py

may have a maximum of one `lemon:canonicalForm` property. Yet, the validator fails if the instance has no `lemon:canonicalForm`, thus suggesting that instead of the `owl:maxCardinality`, a `owl:cardinality` restriction was intended in this case. These kind of semantic subtleties are usually very hard to detect in the complex domain of ontology engineering. It shows that the intensive engagement necessary to write the test cases already serves to debug the ontologies underlying the datasets. This extends the test-driven approach to the ontology development, apart from the quality assessment.

These test cases could directly be translated into SPARQL queries for testing with RDFUnit. For example, it is suggested that a `lemon:LexicalEntry` should contain an `rdfs:label`. As there is no possibility to express these optional constraints in OWL, this test case was added manually to log matching resources as an info-level notice.

Beyond the implementation of the *lemon* validator as test cases, some additional test cases were added to test for semantic correctness or properties that could be added. For example, the `lemon:narrower` relation, which denotes that one sense of a word is narrower than the other, must never be symmetric or contain cycles.

```
SELECT DISTINCT ?s WHERE {
  ?s lemon:narrower+ ?narrower .
  ?narrower lemon:narrower+ ?s . }
```

Similarly, if one resource is `lemon:narrower` to another resource, the inverse relationship (`lemon:broader`) should exist in the database.

From the total of ten manual test cases that were defined for *lemon*, five were described as *PatternBasedTestCase*s, using the existing pattern library, and five as *ManualTestCase*s using custom SPARQL queries. However, for brevity we described the test case with the final SPARQL queries.

NIF

Almost 50% of automated *NIF* test cases were for `rdfs:domain` constraints, 27% for `rdfs:range`, 11% for `owl:FunctionalPropery` restrictions, 7% for disjointness and 5% for proper datatype usage. The early prototype of RDFUnit that is used as the *NIF* validator did not cover any schema constraints and consists of 10 test cases. There exists one test case on the warning level that reports classes of the old namespace.

Other manual test cases include the following restrictions:

- An occurrence of `nif:beginIndex` inside a `nif:Context` must be equal to zero (0).
- The length of `nif:isString` inside a `nif:Context` must be equal to `nif: endIndex`.
- A `nif:anchorOf` string must match the substring of the `nif:isString` from `nif:beginIndex` to `nif:endIndex`. For example:

```
SELECT DISTINCT ?s WHERE  {
  ?s  nif:anchorOf ?anchorOf ;
      nif:beginIndex ?beginIndex ;
      nif:endIndex ?endIndex  ;
      nif:referenceContext [ nif:isString ?referenceString ] .
  BIND (SUBSTR(?referenceString, ?beginIndex ,
               (?endIndex - ?beginIndex) ) AS ?test ) .
  FILTER (str(?test) != str(?anchorOf )) . }
```

- `nif:CString` is an abstract class and thus a subclass such as `nif:CStringImpl` or `nif:RFC5147String` must be used.
- All instances of `nif:CString` that are not `nif:Context` must have a `nif: referenceContext` property.
- All instances of `nif:Context` must also be instances of a `nif:CString` subclass.
- Misspelled `rdf:type` declarations for class names, for example `nif: RFC5147String`.
- All instances of `nif:CString` must have the properties `nif:beginIndex` and `nif:endIndex`.
- all `nif:Context` must have an explicit `nif:isString`, `nif:isString` can only occur with `nif:Context`.

Evaluation

For evaluation purposes we gathered a representative sample of *lemon* and *NIF* datasets in Table 28. We loaded all the datasets in an open-source edition of Virtuoso server (version 7.0)[37] and ran RDFUnit for each one of them. The results of the dataset evaluation are provided in Table 29.

Looking at the results of Table 29 we observe that manual test cases can be of equal importance to the schema restrictions. Additionally we notice that the *lemon*-based datasets were more erroneous than the *NIF*-based datasets. This may be attributed to the following reasons:

- the *NIF* datasets were smaller in size and, thus, better curated.
- the DBpedia Wiktionary datasets is derived from a crowd-sourced source, which makes it more prone to errors.
- the *lemon* ontology is stricter than the *NIF* ontology.
- (Röder et al., 2014) already used the early prototype of RDFUnit and fixed all data errors found by manual test cases.

All *lemon* datasets failed the info level test case that required at least one and unique `lemon:language` in a `lemon:LexicalEntry`. The existence of a `lemon:subsense` or exactly one `lemon:reference` also failed in all datasets with a high number of violations, except Wordnet that had only 33. Additionally, all datasets had a high number of violation on the `owl:minCardinality` of 1 constraint of `lemon:lexicalForm` on the `lemon:LexicalEntry` class. However, all datasets had the appro-

37 http://virtuoso.openlinksw.com

Name	Description	Ontology	Type
lemon datasets			
LemonUby Wiktionary EN [26] (Eckle-Kohler, McCrae, and Chiarcos, 2014)	Conversion of the English Wiktionary into UBY-LMF model	lemon, UBY-LMF	Dictionary
LemonUby Wiktionary DE [27] (Eckle-Kohler, McCrae, and Chiarcos, 2014)	Conversion of the German Wiktionary into UBY-LMF model	lemon, UBY-LMF	Dictionary
LemonUby Wordnet [28] (Eckle-Kohler, McCrae, and Chiarcos, 2014)	Conversion of the Princeton WordNet 3.0 into UBY-LMF model	lemon, UBY-LMF	WordNet
DBpedia Wiktionary [29] (Hellmann, Brekle, and Auer, 2012)	Conversion of the English Wiktionary into lemon	lemon	Dictionary
QHL [30] (Moran and Brümmer, 2013)	Multilingual translation graph from more than 50 lexicons	lemon	Dictionary
NIF datasets			
Wikilinks[31] (Hellmann et al., 2013a)	sample of 60976 randomly selected phrases linked to Wikipedia articles	NIF	NER
DBpedia Spotlight dataset[32] (Steinmetz, Knuth, and Sack, 2013)	58 manually NE annotated natural language sentences	NIF	NER
KORE 50 evaluation dataset[33] (Steinmetz, Knuth, and Sack, 2013)	50 NE annotated natural language sentences from the AIDA corpus	NIF	NER
News-100[34] (Röder et al., 2014)	100 manually annotated German news articles	NIF	NER
RSS-500[35] (Röder et al., 2014)	500 manually annotated sentences from 1,457 RSS feeds	NIF	NER
Reuters-128[36] (Röder et al., 2014)	128 news articles manually curated	NIF	NER

Table 28: Tested datasets

priate number of `lemon:canonicalForm` properties, which is a sub-property of `lemon:lexicalForm` and invalidates these errors. This constraint of RDFUnit, stems from the fact that transitive sub-property checking is not implemented at the time of writing. Except from the DBpedia Wiktionary dataset, all other *lemon* datasets had many reports of a `lemon:LemonEntry` without a label.

The DBpedia Wiktionary dataset had only five failed test cases. With an addition to the previous three, the dataset returned 163K violations due to the disjointness of the `lemon:LexicalEntry` class with the `lemon:LexicalSense` class constraint and 3.5M violations of miss-

	Size	SC	FL	TO	ER	AErrors	MErrors	MWarn	MInfo
WiktDBp	60M	177	5	-	-	3.7M	7.5M	-	3.6M
WktEN	8M	168	14	-	-	752K	394.8K	-	633.3K
WktDE	2M	170	12	-	-	273.1K	66.3K	-	155.6K
Wordnet	4M	166	16	-	-	257.2K	36	-	257.2K
QHL	3M	170	11	-	1	433.1K	539K	-	538K
Wikilinks	0.6M	91	4	-	1	141.528	21.2K	-	-
News-100	13K	91	2	-	3	3.510	-	-	-
RSS-500	10K	91	2	-	3	3.000	-	-	-
Reuters-128	7K	91	2	-	3	2.016	-	-	-
Spotlight	3K	92	3	-	1	662	68	-	-
KORE50	2K	89	6	-	1	301	55	-	-

Table 29: Overview of the NLP datasets test execution. For every dataset, we provide the size in triples count, the number of test cases that were successful, failed, timed-out and did not complete due to an error. Additionally, we mention the total number of the individual violations from automated test cases along with errors, warnings and infos from manual test cases.

ing a required `lemon:lexicalForm` property in a `lemon:LexicalEntry`. The same query returned 270K errors in the QHL dataset.

The Uby Wiktionaries had many failed test cases with a very low (less than 10) number of violations except from `owl:minCardinality` of one in `lemon:Form` class for the `lemon:representation` property. This test case returned 430K errors on the English version and 200K errors on the German. Wordnet also failed this test case with 130K violations. Finally, other high in number of violation test cases are found in the QHL dataset and regard incorrect domain (30K) and range (68K) of `lemon:entry` and wrong range of `lemon:sense` (67K).

The most common test case that failed in all *NIF* datasets is the incorrect datatype of `nif:beginIndex` and `nif:endIndex`. Both properties are defined as `xsd:nonNegativeInteger` but were used as string Literals. This is due to a recent change of the *NIF* specification but also showcases the usefulness of our methodology for data evolution. The correct datatype of `nif:beginIndex` and `nif:endIndex` are also the reason for the *NIF* test cases that returned an error. In these cases, substrings based on these properties were calculated on the query (cf. Section 9.4.2) and non-numeric values did not allow a proper SPARQL query evaluation. This case also expresses the need for chained test cases execution (*TestCaseDependency* in cf. Section 8.2). The existence of a `nif:beginIndex` and `nif:endIndex` in a `nif:CString` also return violation in spotlight (68) kore50 (51) and Wikilinks (21K) datasets. Finally 21K objects in a `nif:wasConvertedFrom` relation did not have `nif:String` as range.

A direct comparison or our results with the results of the implemented validators cannot be provided in a consistent way. The *NIF* validator contained only 10 test cases while our approach had a total of 96 test cases. The *lemon* validator on the other hand could not finish after 48 hours for the DBpedia Wiktionay dataset and resulted in a multitude of non-RDF logging messages that were hard to filter and aggregate.

CONCLUSION AND FUTURE WORK

In this chapter we extensively evaluated the *Test-Driven Quality Assessment* methodology that was described in Chapter 8 using the RDFUnit software tool.

We compiled a comprehensive set of generic Data Quality Test Patterns (DQTP), which we instantiated for 297 schemas resulting in 32,293 test cases. We reused these test cases to evaluate the quality of five LOD datasets. Our evaluation showed that DQTPs are able to reveal a substantial amount of data quality issues in an effective and efficient way.

In addition, we devised 277 test cases for NLP datasets using the *Lemon* and *NIF* vocabularies. We run these test cases on 11 datasets using those vocabularies and containing approximately 23 million triples and identified many millions of errors. We showed progress in implementing domain-specific validation by quickly improving existing validation provided by ontology maintainers

As future work, we aim to extend the test cases to more NLP ontologies, such as MARL, NERD and ITSRDF. We also plan to further increase the scope of the framework, e.g. for the recently changed namespaces of *NIF* and *lemon* deprecation warnings should be produced. Another extension is the modeling of dependencies between test cases, which is currently done manually and could be automated. Furthermore, we also want to apply our methods on services: Usually, semantically enriched NLP services use text as input and return annotations in RDF, which could then be verified by RDFUnit to validate their output.

Additionally, we aim to tackle automatic repair strategies, i.e. use of templates and bindings to fix problems efficiently. We also plan to implement a test-driven data quality cockpit, which allows users to easily instantiate and run DQTPs based on custom knowledge bases.

10 CROWDSOURCING QUALITY IN RDF

Auer and Kontokostas, (2014), Kontokostas, Zaveri, Auer, and Lehmann, (2013), and Zaveri, Kontokostas, Sherif, Bühmann, Morsey, Auer, and Lehmann, (2013)

A key success factor for the Web as a whole was and is its participatory nature – in a truly distributed and democratic nature everybody can publish information on the Web and interlink it with related content. In order to make the Web more intelligent, we have to empower people to easily create and interlink structured information and semantics. Also, rich semantics can probably not emerge only by automated means. In physics, the law of conservation of energy states that the total energy of an isolated system cannot change – it is said to be conserved over time. A similar observation seems to hold for semantics: rich semantics can not appear miraculously out of nowhere, even if the most advanced and powerful algorithms are applied. In fact, two main strategies for 'increasing the intelligence' of the Web seem to be currently prevalent:

- Translating non-machine readable semantics into machine-readable semantics – this is, for example, what most natural language processing techniques aim at.
- Enriching existing semantic representations with additional or related information, for example, through link discovery.

However, the most progress can be made, if we can put the human intelligence in the loop. There were a number of strategies aiming to achieve that: games with a purpose, semantic (data) wikis and crowd-sourcing. Of course these strategies are not strictly separable, but overlap in many aspects. Semantic Wikis (Frischmuth et al., 2015; Krötzsch et al., 2007) apply the wiki paradigm of 'making it easy to add and change information' to structured knowledge-bases. Small bits and pieces of information can be added to the semantic wiki in an agile manner. In this regard, semantic wikis can be considered as one technology to facilitate crowd-sourcing. However, other than voluntary contributions to a wiki, crowd-sourcing often involves monetary incentives and large groups of crowd-workers aiming to achieve a specified goal in short time. Games with a purpose (or human-based computation games) (Siorpaes and Hepp, 2008) on the other hand also implement some kind of crowd-sourcing, where the monetary incentive is replaced by some entertainment and recognition.

On the Data Web, we have varying quality of information covering various domains. There are a large number of high quality datasets (in particular in the life-sciences domain), which are carefully curated over decades and recently published on the Web. There are, however, also many datasets, which were extracted from unstructured and

semi-structured information or are the result of some crowdsourcing process, where large numbers of users contribute small parts. DBpedia is actually an example for both - a dataset extracted from the result of a crowdsourcing process. Hence, quality problems are inherent in DBpedia. This is not a problem per se, since quality usually means fitness for a certain use case (Juran, 1974). Hence, even datasets with quality problems might be useful for certain applications, as long as the quality is in the required range.

In this chapter, we devise a data quality assessment methodology, which comprises of a manual process. We empirically assess, based on this methodology, the data quality of one of the major knowledge hubs on the Data Web – DBpedia. The first phase includes the detection of common quality problems and their representation in a comprehensive taxonomy of potential quality problems. In the manual process, the second phase comprises of the evaluation of a large number of individual resources, according to the quality problem taxonomy, using *crowdsourcing* in order to evaluate the type and extent of data quality problems occurring in DBpedia. Each represented fact is evaluated for correctness by each user and, if found problematic, annotated with one of 17 pre-defined quality criteria. This process is accompanied by a tool wherein a user assesses an individual resource and evaluates each fact for correctness.

We find that while a substantial number of problems exists, the overall quality is with a less than 11.93% error rate relatively high. With this study we not only aim to assess the quality of DBpedia but also to adopt a methodology to improve the quality in future versions by regularly providing feedback to the DBpedia maintainers to fix these problems.

Our main contributions are:

- a crowdsourcing based methodology for data quality assessment (Section 10.1),
- a crowdsourcing based data quality assessment tool (Section 10.2),
- an empirical data quality analysis of the DBpedia dataset performed using crowdsourcing (Section 10.3) and

We conclude with an outlook on future work in Section 10.4.

ASSESSMENT METHODOLOGY

In this section, we describe a generalized methodology for the assessment and subsequent data quality improvement of resources belonging to a dataset. The assessment methodology we propose is depicted in Figure 27. This methodology consists of the following four steps: 1. Resource selection, 2. Evaluation mode selection, 3. Resource evaluation and 4. Data quality improvement. In the following, we describe these steps in more detail.

STEP I: RESOURCE SELECTION In this first step, the resources belonging to a particular dataset are selected. This selection can be performed in three different ways:

- *Per Class:* select resources belonging to a particular class
- *Completely random:* a random resource from the dataset
- *Manual:* a resource selected manually from the dataset

Choosing resources per class (e.g. animal, sport, place etc.) gives the user the flexibility to choose resources belonging to only those classes she is familiar with. However, when choosing resources from a class, the selection should be made in proportion to the number of instances of that class. Random selection, on the other hand, ensures an unbiased and uniform coverage of the underlying dataset. In the manual selection option, the user is free to select resources with problems that she has perhaps previously identified.

STEP II: EVALUATION MODE SELECTION The assignment of the resources to a person or machine, selected in Step I, can be accomplished in the following three ways:

- *Manual:* the selected resources are assigned to a person (or group of individuals) who will then proceed to manually evaluate the resources individually.
- *Semi-automatic:* selected resources are assigned to a semi-automatic tool which performs data quality assessment employing some form of user feedback.
- *Automatic:* the selected resources are given as input to an automatic tool which performs the quality assessment without any user involvement.

STEP III: RESOURCE EVALUATION The person (or group of individuals) evaluates each resource individually to detect the potential data quality problems. In order to support this step, a quality assessment tool can be used which allows a user to evaluate each individual triple belonging to a particular resource. If, in case of Step II, the selected resources are assigned to a semi-automatic tool, the tool points to triples likely to be wrong. For example, domain or range problems are identified by the tool and then assigned to a person to verify the correctness of the results.

STEP IV: DATA QUALITY IMPROVEMENT After the evaluation of resources and identification of potential quality problems, the next step is to improve the data quality. There are at least two ways to perform an improvement:

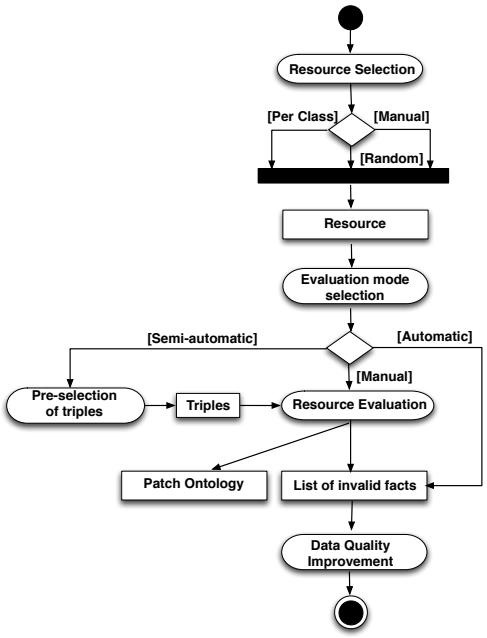


Figure 27: Workflow of the data quality assessment methodology.

- Direct: editing the triple, identified to contain the problem, with the correct value
- Indirect: using the Patch Request Ontology[1] (Knuth, Hercher, and Sack, 2012) which allows gathering user feedbacks about erroneous triples.

A systematic review done in (Zaveri et al., 2015) identified a number of different data quality dimensions (criteria) applicable to Linked Data. After carrying out an initial data quality assessment on DBpedia (as part of the first phase of the manual assessment methodology), the problems identified were mapped to this list of identified dimensions. In particular, *Accuracy, Relevancy, Representational-consistency and Interlinking* were identified to be problems affecting a large number of DBpedia resources. Additionally, these dimensions were further divided into categories and sub-categories.

Table 30 gives an overview of these data quality dimensions along with their categories and sub-categories. Moreover, the table specifies whether the problems are specific to DBpedia (marked with a ✔) or could potentially occur in any RDF dataset. For example, the sub-category *Special template not properly recognized* is a problem that occurs only in DBpedia due to the presence of specific keywords in Wikipedia articles that do not cite any references or resources (e.g. {{Unreferenced stub|auto=yes}}). On the other hand, the problems that are not DBpedia specific can occur in any other datasets. We

1 http://141.89.225.43/patchr/ontologies/patchr.ttl#

Dimension	Category	Sub-category	DBpedia specific
Accuracy	Triple incorrectly extracted	Object value is incorrectly extracted	–
		Object value is incompletely extracted	–
		Special template not properly recognized	✔
	Datatype problems	Datatype incorrectly extracted	–
	Implicit relationship between attributes	One fact encoded in several attributes	✔
		Several facts encoded in one attribute	–
		Attribute value computed from another attribute value	✔
Relevancy	Irrelevant information extracted	Extraction of attributes containing layout information	✔
		Redundant attribute values	–
		Image related information	✔
		Other irrelevant information	–
Represensational-Consistency	Representation of number values	Inconsistency in representation of number values	–
Interlinking	External links	External websites	–
	Interlinks with other datasets	Links to Wikimedia	–
		Links to Freebase	–
		Links to Geospecies	–
		Links generated via Flickr wrapper	–

Table 30: Data quality dimensions, categories and sub-categories identified in the DBpedia resources. The DBpedia specific column denotes whether the problem type is specific only to DBpedia (tick) or could occur in any RDF dataset.

refer the reader to (Zaveri et al., 2013) for further analysis on the dimensions and metrics listed on Table 30.

TRIPLECHECKMATE: A CROWDSOURCING QUALITY ASSESSMENT TOOL

TripleCheckMate is a tool built specifically for the purpose of the Quality Assessment Methodology (cf. Section 10.1). The tool is released as open source[2] under the Apache License.

In the following, we describe TripleCheckMate from a user perspective (Section 10.2.1) as well as the system architecture (Section 10.2.2) and extensibility of the tool (Section 10.2.3).

2 https://github.com/AKSW/TripleCheckMate

Overview

The design of TripleCheckMate is aligned with the methodology described in Section 10.1, in particular with Steps 1–3. To use the tool, the user is required to authenticate herself, which not only prevents spam but also helps in keeping track of her evaluations. After authenticating herself, she proceeds with the selection of a resource (Step 1). She is provided with three options: (i) *per class*, (ii) *completely random* and (iii) *manual* (as described in Step I of the assessment methodology).

After selecting a resource, the user is presented with a table showing each triple belonging to that resource on a single row. Step 2 involves the user evaluating each triple and checking whether it contains a data quality problem. The link to the original Wikipedia page for the chosen resource is provided on top of the page which facilitates the user to check against the original values. If the triple contains a problem, she checks the box "is wrong". Moreover, she is provided with a taxonomy of pre-defined data quality problems (cf. Table 30) where she assigns each incorrect triple to a problem. If the detected problem does not match any of the existing types, she has the option to provide a new type and extend the taxonomy. After evaluating one resource, the user saves the evaluation and proceeds to choosing another random resource and follow the same procedure.

An important feature of the tool is to allow measuring of inter-rater agreements. This means, when a user selects a random method (*Any* or *Class*) to choose a resource, there is a 50% probability that she is presented with a resource that was already evaluated by another user. This probability as well as the number of evaluations per resource is configurable. Allowing many users evaluating a single resource not only helps to determine whether incorrect triples are recognized correctly but also to determine incorrect evaluations (e.g. incorrect classification of problem type or marking correct triples as incorrect), especially when crowdsourcing the quality assessment of resources.

Architecture

TripleCheckMate is built with *Java* using the *Google Web Toolkit*[5] (GWT) development toolkit. GWT allows one to build and optimize complex browser-based applications as it provides a static typed programming interface (Java) and compiles the output to native cross-browser HTML+Javascript. A Java Web Server (Apache Tomcat or Jetty) is used as a backend along with a MySQL database engine.

4 http://nl.dbpedia.org:8080/TripleCheckMate-Demo
4 http://www.youtube.com/watch?feature=player_embedded&v=l-StthTvjFI
5 https://developers.google.com/web-toolkit/

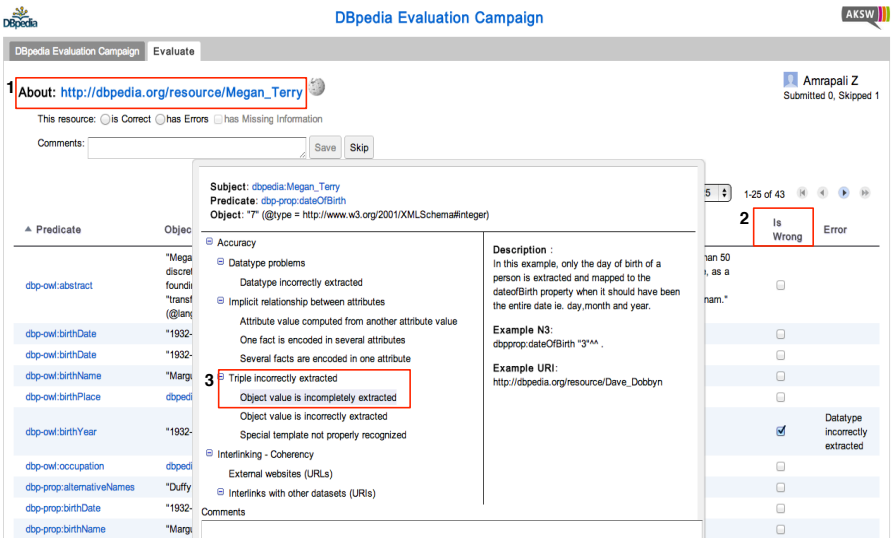


Figure 28: Screenshot of the TripleCheckMate data quality assessment tool. First, a user chooses a resource. Second, she is displayed with all triples belonging to that resources and evaluates each triple individually to detect quality problems. Third, If she finds a problem, she marks it and associates it with a relevant problem category. A demo[3] and a screencast[4] of the tool are available.

Figure 29 depicts the general TripleCheckMate architecture. In order to minimize the dependencies and make TripleCheckMate as portable as possible, all the application logic is built on the frontend. The applications' backend is used only to store and retrieve evaluation related data. The database schema of the backend is depicted in Figure 30.

When the user enters the application, the available campaign and all the relevant configuration is loaded. In the future we plan to support multiple simultaneous campaigns. The user authentication is performed through the *Google OAuth 2.0*[6] protocol. If the user enters the application for the first time, her entry is transparently stored in the users table. A new session is created every time a user enters the application and all the evaluations in that session are associated. In order to speed-up the *per class* resource selection option, the class hierarchy is cached in the *Dataset ontology* table.

Finally, our data model separates the general resource evaluation (*Assessed resources*) from the detailed triple evaluation (*Assessed resource triples*). The rationale for this are cases where the user wants to mark a resource as completely correct or comment on missing information. In *Assessed resource triples* we store detailed evaluations at the triple level and assign errors to triples based on the *Quality taxonomy*.

Technically, after every complete resource evaluation, the evaluation results are submitted to the server and the user statistics are

6 https://developers.google.com/accounts/docs/OAuth2

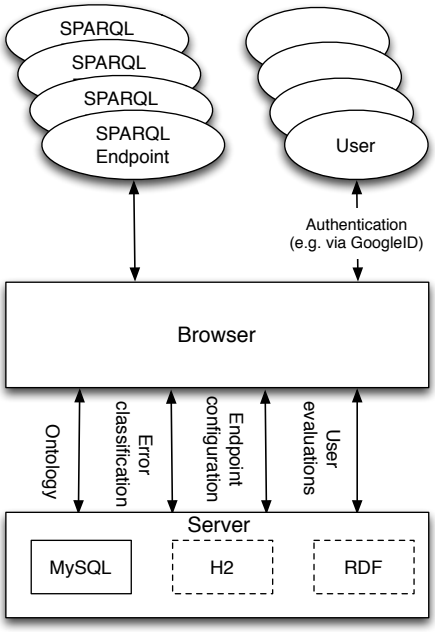


Figure 29: Architecture of the TripleCheckMate tool.

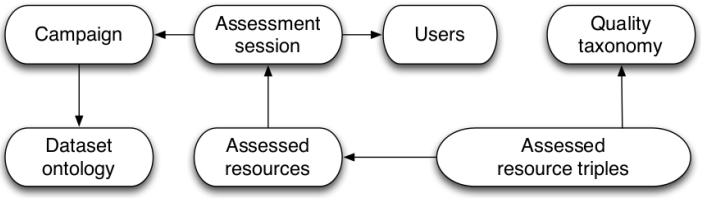


Figure 30: The backend database schema where the arrows depict the foreign key constrains relations between the tables.

re-aggregated for an up-to-date contributor ranking. The communication to the server is performed with RPC requests facilitated by the GWT developer toolkit. Finally, for the SPARQL Endpoint communication we implemented a very lightweight client-side SPARQL framework where we encode SPARQL queries in *GET* requests and parse results in the JSON-LD format.

Extensibility

Although TripleCheckMate was initially built for the purpose of the *DBpedia Evaluation Campaign*, it can meanwhile be easily customized to support any arbitrary (open or closed) dataset with a SPARQL endpoint.

Most of the configurations lie in the database. The SPARQL endpoint configuration is stored in the *campaign* table. The Dataset ontology and *Quality Taxonomy* hold the respective data as tree structures and, thus, can be easily changed to any ontology or taxonomy. The

database connection configuration is stored in property files. Finally, visual end-user changes can be performed directly into the HTML and GWT Layout files.

As a data store backend, we support MySQL at the moment, although any JDBC compatible database can be used. The embedded in-memory H2 database is also supported as it provides a JDBC interface. We plan to ship TripleCheckMate with a preconfigured H2 database, as a standalone evaluation tool. Finally, RDF as data store is also a feature our community base expressed interest to implement.[7]

7 https://groups.google.com/d/topic/dbpedia-data-quality/rkXfR1BR4uY/discussion

EVALUATION OF DBPEDIA DATA QUALITY

Evaluation Methodology

We performed the assessment of the quality of DBpedia in two phases: *Phase I: Problem detection and creation of taxonomy* and *Phase II: Evaluation via crowdsourcing*.

Phase I: Creation of quality problem taxonomy. In the first phase, two researchers independently assessed the quality of 20 DBpedia resources each. During this phase an initial list of data quality problems, that occurred in each resource, was identified. These identified problems were mapped to the different quality dimensions from (Zaveri et al., 2015). After analyzing the root cause of these problems, a refinement of the quality dimensions was done to obtain a finer classification of the dimensions. This classification of the dimensions into subcategories resulted in a total of 17 types of data quality problems (cf. Table 30).

Phase II: Crowdsourcing quality assessment. In the second phase, we crowdsourced the quality evaluation wherein we invited researchers who are familiar with RDF to use the TripleCheckMate tool (described in Section 10.2). First, each user after authenticating oneself, chooses a resource by one of three options mentioned in Section 10.1. Thereafter, the extracted facts about that resource are shown to the user. The user then looks at each individual fact and records whether it contains a data quality problem and maps it to the type of quality problem.

Evaluation Results

An overview of the evaluation results is shown in Table 31[8]. Overall, only 16.5% of all resources were not affected by any problems. On average, there were 5.69 problems per resource and 2.24 problems excluding errors in the *dbprop namespace*[9] (Lehmann et al., 2009). While the vast majority of resources have problems, it should also be remarked that each resource has 47.19 triples on average, which is higher than in most other LOD datasets. The tool was configured to allow two evaluations per resource and this resulted to a total of 268 inter-evaluations. We computed the inter-rater agreement for those resources, which were evaluated by two persons by adjusting the observed agreement with agreement by chance as done in Cohen's kappa[10]. The inter-rater agreement results – 0.34 for resource agreement and 0.38 for triple agreement – indicate that the same resource should be evaluated more than twice in future evaluations. To assess

8 Also available at: http://aksw.org/Projects/DBpediaDQ
9 http://dbpedia.org/property/
10 http://en.wikipedia.org/wiki/Cohen%27s_kappa

Total no. of users	58
Total no. of distinct resources evaluated	521
Total no. of resources evaluated	792
Total no. of distinct resources without problems	86
Total no. of distinct resources with problems	435
Total no. of distinct incorrect triples	2928
Total no. of distinct incorrect triples in the *dbprop* namespace	1745
Total no. of inter-evaluations	268
No. of resources with evaluators having different opinions	89
Resource-based inter-rater agreement (Cohen's Kappa)	0.34
Triple-based inter-rater agreement (Cohen's Kappa)	0.38
No. of triples evaluated for correctness	700
No. of triples evaluated to be correct	567
No. of triples evaluated incorrectly	133
% of triples correctly evaluated	81
Average no. of problems per resource	5.69
Average no. of problems per resource in the *dbprop* namespace	3.45
Average no. of triples per resource	47.19
% of triples affected	11.93
% of triples affected in the *dbprop* namespace	7.11

Table 31: Overview of the manual quality evaluation.

the accuracy of the crowdsourcing evaluation, we took a random sample of 700 assessed triples (out of the total 2928) and evaluated them for correctness based on the formula in (Krejcie and Morgan, 1970) intended to be a representative of all the assessed triples. Additionally, we assumed a margin of 3.5% of error, which is a bound that we can place on the difference between the estimated correctness of the triples and the true value, and a 95% confidence level, which is the measure of how confident we are in that margin of error[11]. From these 700 triples, 133 were evaluated incorrectly resulting in about 81% of triples correctly evaluated.

Table 32 shows the total number of problems, the distinct resources and the percentage of affected triples for each problem type. Overall, the most prevalent problems, such as broken external links are outside the control of the DBpedia extraction framework. After that, several extraction and mapping problems that occur frequently mainly affecting accuracy, can be improved by manually adding mappings or possibly by improving the extraction framework.

When looking at the detectable and fixable problems from Table 30, in light of their prevalence, we expect that approximately one third of the problems can be automatically detected and two thirds are fixable by improving the DBpedia extraction framework. In particular,

11 http://research-advisors.com/tools/SampleSize.htm

Criteria	IT	DR	AT %
Accuracy			
Object incorrectly extracted	32	14	2.69
Object value is incorrectly extracted	259	121	23.22
Object value is incompletely extracted	229	109	20.92
Special template not recognized	14	12	2.30
Datatype problems	7	6	1.15
Datatype incorrectly extracted	356	131	25.14
Implicit relationship between attributes	8	4	0.77
One fact is encoded in several attributes	670	134	25.72
Several facts encoded in one attribute	87	54	10.36
Value computed from another value	14	14	2.69
Accuracy unassigned	31	11	2.11
Relevancy			
Irrelevant information extracted	204	29	5.57
Extraction of layout information	165	97	18.62
Redundant attributes value	198	64	12.28
Image related information	121	60	11.52
Other irrelevant information	110	44	8.45
Relevancy unassigned	1	1	0.19
Representational-consistency			
Representation of number values	29	8	1.54
Representational-consistency unassigned	5	2	0.38
Interlinking			
External websites (URLs)	222	100	19.19
Interlinks with other datasets (URIs)	2	2	0.38
Links to Wikimedia	138	71	13.63
Links to Freebase	99	99	19.00
Links to Geospecies	0	0	0.00
Links generated via Flickr wrapper	135	135	25.91
Interlinking unassigned	3	3	0.58

Table 32: Detected number of problem for each of the defined quality problems. IT = Incorrect triples, DR = Distinct resources, AT = Affected triples.

implicitly related attributes can be properly extracted with a new extractor, which can be configured using the DBpedia Mappings Wiki. As a result, we expect that the improvement potential is that the problem rate in DBpedia can be reduced from 11.93% to 5.81% (calculated by subtracting 7.11% from 11.93% reported in Table 31). After revising the DBpedia extraction framework, we will perform subsequent quality assessments using the same methodology in order to realize and demonstrate these improvements.

CONCLUSIONS AND FUTURE WORK

To the best of our knowledge, this study is the first comprehensive empirical quality analysis for more than 500 resources of a large Linked Data dataset extracted from crowdsourced content. We found that a substantial number of problems exist and the overall quality, with a 11.93% error rate, is moderate. In addition to the quality analysis of DBpedia, we devised a generic methodology for Linked Data quality analysis, derived a comprehensive taxonomy of extraction quality problems and developed a tool which can assist in the evaluation. All these contributions can be reused for analyzing any other extracted dataset (by domain experts). The detailed analysis of data quality problems allows us to devise and implement corresponding mitigation strategies. Many of the problems found can be firstly automatically detected and secondly avoided by (1) improving existing extractors, (2) developing new ones (e.g. for implicitly related attributes) or (3) improving and extending mappings and extraction hints on the DBpedia Mappings Wiki.

With this study, we not only aim to assess the quality of this sample of DBpedia resources but also adopt an agile methodology to improve the quality in future versions by regularly providing feedback to the DBpedia maintainers to fix these problems. We plan to improve the DBpedia extraction framework along these detected problems and periodically revisit the quality analysis (in regular intervals) in order to demonstrate possible improvements.

With regard to TripleCheckMate, it is a tool that has already been successfully tested in assessing the quality of DBpedia and can be easily configured to work with any dataset that provides a SPARQL endpoint. In future versions of the tool, we will include further support for the methodology outlined in Section 10.1 by directly integrating semi-automatic methods, which can then filter those triples of a resource, which are most likely to cause problems. We will investigate whether this can improve the efficiency of the quality assessment. Moreover, we also plan to include support for the patch ontology (Knuth, Hercher, and Sack, 2012) as an output format.

11

RELATED WORK ON RDF AND LINKED DATA QUALITY

Kontokostas, Brümmer, Hellmann, Lehmann, and Ioannidis, (2014b,c) and Zaveri, Kontokostas, Sherif, Bühmann, Morsey, Auer, and Lehmann, (2013)

This chapter provides the related work on RDF and Linked data quality with regard to Part iii and Part iv of this thesis.

STANDARDS ON QUALITY ASSESSMENT

SPARQL *Inferencing Notation* (SPIN) (Knublauch, Hendler, and Idehen, 2011) is a W3C submission aiming at representing rules and constraints on Semantic Web models. SPIN also allows users to define SPARQL functions and reuse SPARQL queries. The difference between SPIN and the *Test-Driven Quality Assessment Methodology* (TDQAM) is that SPIN functions would not fully support our *Pattern Bindings*. SPIN function arguments must have specific constraints on the argument datatype or argument class and do not support operators, e.g. '=', '>', '!', '+', '*', or property paths[1]. However, TDQAM is still compatible with SPIN when allowing to initialise templates with specific sets of applicable operators. In that case, however, the number of templates increases. Due to this restrictions, SPIN defines fewer but more general constraints. The following SPIN example[2] tries to locate all the `owl:disjointWith` constraint violations:

```
1 SELECT ?x WHERE { ?c1 owl:disjointWith ?c2 .
2                   ?x a ?c1 .
3                   ?x a ?c2 . }
```

The problems of these types of queries is that: 1) they are more expensive to execute, 2) aggregate all errors in a single result which makes it harder to debug and 3) cannot capture violations like `foaf:primaryTopic` if the `foaf` schema is not loaded in the knowledge base itself. One of the advantages of converting our templates to SPIN is that the structure of the SPARQL query itself can be stored directly in RDF, which, however, renders it more complex. From the efforts related to SPIN, TDQAM reuses their existing data quality patterns and ontologies for error types.

IBM Resource Shapes (RS)[3] is a W3C member submission that is based on based on the Open Services for Lifecycle Collaboration (OSLC) Core specification. [4]. RS defines a high-level RDF vocabulary for specifying the shape of RDF resources. The shape of an RDF re-

1 http://www.w3.org/TR/2010/WD-sparql11-property-paths-20100126/
2 http://topbraid.org/spin/owlrl-all.html#cax-dw
3 https://www.w3.org/Submission/shapes/
4 http://open-services.net/bin/view/Main/OslcCoreSpecification

source is a description of the set of triples it is expected to contain and the integrity constraints those triples are required to satisfy. Applications of shapes include validating RDF data, documenting RDF APIs, and providing metadata to tools, such as form and query builders, that handle RDF data. The TDQA methodology is compatible with RS and *Test Auto Generators* were defined that consumed a big subset of RS definitions in RDFUnit.[5]

Shape Expressions (ShEx) (Prud'hommeaux, Labra Gayo, and Solbrig, 2014) is a language for describing RDF graph structures. ShEx shapes describes the triples touching nodes in RDF graphs. These descriptions identify predicates and their associated cardinalities and datatypes. ShEx shapes can be used to communicate data structures associated with some process or interface, generate or validate data, or drive user interfaces.

SPIN, RS and ShEx share the same goals but offer different tradeoffs with regard to constraint expressivity, verboseness and simplicity. SPIN is the most powerful in terms of constraint expressivity. RS covers common use cases in a simple and intuitive way but it is based in RDF and is more verbose than ShEx. ShEx on the other hand offers a custom compact syntax tailored for the ShEx language and provides a little better constraint expresivity than RS.

SHACL (Knublauch and Kontokostas, 2016) is the upcoming W3C standard that consumes and merges all these languages in a new & powerful constraint language. SHACL will feature shapes as a basic validation component that is inspired from RS and ShEx. SHACL offers an extensive set of core pre-defined constrains and an extension mechanism that is based in SPARQL, similar to SPIN. A compact syntax, similar to ShEx will be created on top of SHACL. SHACL additionally borrowed a lot of features from TDQAM and RDFUnit. Naming a few is constraint severity levels, SPARQL constraint definitions based on SELECT queries and SPARQL variable/value bindings.

WEB DATA QUALITY ASSESSMENT FRAMEWORKS

There are a number of data quality assessment dimensions that have already been identified relevant to Linked Data, namely, accuracy, timeliness, completeness, relevancy, conciseness, consistency, to name a few (Bizer, 2007). Additional quality criteria such as uniformity, versatility, comprehensibility, amount of data, validity, licensing, accessibility and performance were also introduced to be additional means of assessing the quality of LOD (Flemming, 2010). Additionally, there are several efforts in developing data quality assessment frameworks in order to assess the data quality of LOD. These efforts are either semi-automated (Flemming, 2010), automated (Guéret et al., 2012) or manual (Bizer and Cyganiak, 2009; Mendes P.N., 2012).

5 https://github.com/AKSW/RDFUnit/issues/23

Even though these frameworks introduce useful methodologies to assess the quality of a dataset, either the results are difficult to interpret, do not allow a user to choose the input dataset or require a considerable amount of user involvement.

The LUZZU Quality Assessment Framework [6] (Debattista, Lange, and Auer, 2014) is a generic framework based on the Dataset Quality Ontology (daQ)[7], allowing users to define their own quality metrics. Luzzu assesses Linked Data quality, using a library of generic and user-provided domain specific quality metrics, provides queryable quality metadata and assembles detailed quality reports. However, LUZZU's stream processing architecture can evaluate quality metrics that relate to a single RDF triple.

CONCRETE WEB DATA QUALITY ASSESSMENTS

An effort to assess the quality of web data was undertaken in 2008 (Cafarella et al., 2008), where 14.1 billion HTML tables from Google's general-purpose web crawl were analyzed in order to retrieve those tables that have high-quality relations. Additionally, there have been studies focused on assessing the quality of RDF data (Hogan et al., 2010) to report the errors occurring while publishing RDF data and the effects and means to improve the quality of structured data on the web. As part of an empirical study (Hogan et al., 2012) 4 million RDF/XML documents were analyzed, which provided insights into the level of conformance in these documents with respect to the Linked Data guidelines. Even though these studies accessed a vast amount of web or RDF/XML data, most of the analysis was performed automatically and therefore the problems arising due to contextual discrepancies were overlooked. Another study aimed to develop a framework for the DBpedia quality assessment (Kreis, 2011). In this study, particular problems of the DBpedia extraction framework were taken into account and integrated in the framework. However, only a small sample (75 resources) were assessed in this case and an older DBpedia version (2010) was analyzed.

CROWDSOURCING-BASED TASKS

There are already a number of efforts which use crowdsourcing focused on a specific type of task. For example, crowdsourcing is used for entity linking or resolution (Demartini, Difallah, and Cudré-Mauroux, 2012), quality assurance and resource mangement (Wang et al., 2012) or for enhancement of ontology alignments (Sarasua, Simperl, and Noy, 2012) especially in Linked Data. However, in our case (Chapter 10), we originally did not submit tasks to the popular internet

6 http://eis-bonn.github.io/Luzzu/
7 http://purl.org/eis/vocab/daq

marketplaces such as Amazon Mechanical Turk or CrowdFlower[8]. Instead, we used the intelligence of a large number of researchers who were particularly conversant with RDF to help assess the quality of one of the important and most linked dataset, DBpedia. A combination of crowdsourcing by both domain experts and simple workers, in combination of automated approaches like TDQAM is performed in (Acosta et al., 2013) with positive results.

PREVIOUS DATA QUALITY ASSESSMENTS ON DBPEDIA

The first publication of DBpedia (Auer and Lehmann, 2007) mainly concentrates on the data source – Wikipedia. Errors in the RDF data are attributed to several shortcomings in the authoring process, e.g. the usage of tables instead of templates, the encoding of layout information like color in templates and so on. Other inaccuracies occur due to an imprecise use of the wiki markup or when duplicate information is given, as in *height = 5'11" (180cm)*. To avoid those errors the authors provide some authoring guidelines in accordance with guidelines created by the Wikipedia community.

In Lehmann et al., (2009), the authors concentrate more on the extraction process, comparing the *Generic* with the *Mapping-based Infobox Extraction* approach. It is shown that by mapping Wikipedia templates to a manually created, simple ontology, one can obtain a far better data quality, eliminating data type errors as well as a better linkage between entities of the dataset. Other errors concern class hierarchies e.g. omissions in the automatically created YAGO classification schema.

Another issue already addressed in the future work section of Lehmann et al., (2009) is the fusion of cross-language knowledge of the language specific DBpedia instances. This topic as well as other internationalization issues are treated in Kontokostas et al., (2012). There, different extraction problems of the Greek DBpedia are presented that can also be applied to other languages, especially those using non-Latin characters.

Another study aimed to develop a framework for the DBpedia quality assessment is presented in Chapter 10 and Zaveri et al., (2013) and involves a manual and a semi-automatic process. In the manual phase the authors detects common problems and classify them in a taxonomy. After that, they crowdsource the evaluation of a large number of individual resources and let users structure it according to their taxonomy. This work is extended in (Acosta et al., 2013) by including non-domain experts in the crowdsourcing tasks, as well as including automatied assessment methods.

In (Färber et al., 2016), the authors try to provide an extensive comparison, with regard to quality of DBpedia, Freebase, OpenCyc, Wiki-

8 http://crowdflower.com/

data and YAGO. Finally, (Paulheim, 2016) provides a survey of evaluation methods on knowledge graphs, including DBpedia.

RULES AND SPARQL

The approach described in (Fürber and Hepp, 2010) advocates the use of SPARQL and SPIN for RDF data quality assessment and shares some similarity with our methodology. However, a domain expert is required for the instantiation of test case patterns. In a similar way, Fürber et al. (Fürber and Hepp, 2010) define a set of generic SPARQL queries to identify missing or illegal literal values and datatypes and functional dependency violations.

Another related approach is the *Pellet Integrity Constraint Validator* (ICV)[9]. *Pellet ICV* (Sirin and Tao, 2009) translates OWL integrity constraints into SPARQL queries. Similar to our approach, the execution of those SPARQL queries indicate violations. An implication of the integrity constraint semantics of Pellet ICV is that a partial unique names assumption (all resources are considered to be different unless equality is explicitly stated) and a closed world assumption is in effect. We use the same strategy as part of our methodology, but go beyond it by allowing users to directly (re-)use DQTPs not necessarily encoded in OWL and by providing automatic schema enrichment.

Schemarama[10] is a very early (2001) constraint validation approach based on using the Squish RDF language instead of SPARQL. It does not offer a templating mechanism or a classification of data quality problems. For XML, *Schematron*[11] is an ISO standard for validation and quality control of XML documents based on XPath and XSLT. We argue that similar adapted mechanisms for RDF are of crucial importance to provide solutions allowing the usage of RDF in settings, which require either high quality data or at least an accurate assessment of its quality.

In database research, there are related approaches to formulate common integrity constraints (Deutsch, 2009) using First Order Logic (FOL). The work presented in (Fan, 2008) uses FOL to describe data dependencies for quality assessment and suggests repairing strategies. Finally, in (Lausen, Meier, and Schmidt, 2008), the authors suggest extensions to RDF by constraints akin to RDBMS in order to validate data using SPARQL as a constraint language. This is achieved by providing an RDF view on top of the data.

9 http://clarkparsia.com/pellet/icv/
10 http://swordfish.rdfweb.org/discovery/2001/01/schemarama/
11 http://www.schematron.com/

Part IV

TEST-DRIVEN QUALITY ASSESSMENT IN OTHER DOMAINS

12

ASSESSING AND REFINING MAPPINGS TO RDF TO IMPROVE DATASET QUALITY

Dimou, Kontokostas, Freudenberg, Verborgh, Lehmann, Mannens, Hellmann, and Walle, (2015)

The Linked Open Data (LOD) cloud[1] consisted of 12 datasets in 2007, grew to almost 300 in 2011[2], and, by the end of 2014, counted up to 1,100[3]. Although more and more data is published as Linked Data (LD), the datasets' quality and consistency varies significantly, ranging from expensively curated to relatively low quality datasets (Zaveri et al., 2015). In Chapter 9, we observed that similar violations can occur very frequently. Especially when datasets stem originally from semi-structured formats (csv, XML, etc.) and their RDF representation is obtained by repetitively applying certain mappings, the violations are often repeated, as well. By *mapping*, we consider the function of semantically annotating data to acquire their enriched representation using the RDF data model. A mapping consists of one or more *mapping definitions* (MDs) that state how RDF terms should be generated, taking into account a data fragment from an original data source, and how these terms are associated to each other and form RDF triples.

The most frequent violations are related to the dataset's schema, namely the vocabularies or ontologies used to annotate the original data (Kontokostas et al., 2014c). In the case of semi-structured data, the dataset's schema derives from the set of classes and properties specified within the mappings. A mapping might use a single ontology or vocabulary to annotate the data, or a proprietary vocabulary can be generated as the data is annotated. Lately, combinations of different ontologies and vocabularies are often used to annotate data (Schmachtenberg, Bizer, and Paulheim, 2014), which increases the likelihood of such violations. A violation might derive from (i) *incorrect usage of schemas* in the mapping definitions; and (ii) mistakes in the *original data source*. The second category of violations can be resolved by cleansing the data. In this work, we focus specifically on the first, which is directly related to the mapping process.

Only recently, several research efforts started focusing on formalising LD quality tracking and assessment (Zaveri et al., 2015). Nevertheless, such formalisation approaches remain independent of the LD mapping and publishing process—quality assessment is not even mentioned in the best practices for publishing LD (Hyland, Atemezing, and Villazon-Terrazas, 2004). Existing quality assessment refers to already published data and is, in most cases, performed by third parties rather than data publishers. Thus, incorporating quality as-

1 http://lod-cloud.net/
2 http://lod-cloud.net/state
3 http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/

sessment results corresponds to incorporating a *Linked Data feedback loop*: existing LD infrastructures still neither intuitively process end-users' input, nor properly propagate the modifications to the MDs and original data. Consequently, the results are rarely and, if so, manually used to adjust the dataset, with the risk of being overwritten when a new version of the original data is published.

In this chapter, we therefore propose a methodology that extends LD quality assessment from data *consumption* to also cover data *publication*. We transform the assessment process normally applied to the final dataset so that it applies to the mappings as well. This allows publishers to discover mistakes in mapped RDF data—before they are even generated. Our methodology (i) augments the mapping and publishing workflow of semi-structured source formats with *systematic Linked Data quality assessments* for both the mappings and the resulting dataset; and (ii) *automatically suggests mapping refinements* based on the results of these quality assessments. We consider iterative, uniform, and gradual test-driven LD quality assessments to improve the dataset's overall quality.

The chapter is organized as follows: Section 12.1 details the need of quality assessment during the mapping process, followed by the introduction of a mapping workflow with quality assessment in Section 12.2. Next, Section 12.3 explains how quality assessments are applied to mappings, the results of which are used to refine mapping definitions in Section 12.4. Section 12.5 highlights different cases where the proposed workflow was used, followed by an evaluation in Section 12.6. Finally, Section 12.7 and Section 12.8 summarize related solutions and conclusions.

INCORPORATING QUALITY IN MAPPING AND PUBLISHING

Data quality is commonly conceived as "fitness for use" for a certain application or use case (Juran, 1974). A *data quality assessment metric, measure,* or *indicator* is a procedure for measuring a data quality dimension (Bizer and Cyganiak, 2009). A *data quality assessment methodology* is defined as the process of evaluating whether a piece of data meets the information that consumers need in a specific use case (Bizer and Cyganiak, 2009). In this respect, our use case is focused on the quality of the generated RDF dataset compared to the ontologies and vocabulary definitions of its schema. The uppermost goal is aiding data publishers to finally acquire a valid and high quality LD by annotating semi-structured data. We focus on the *intrinsic* dimension of data quality (Zaveri et al., 2015).

The earlier dataset quality is assessed, the better: we argue that mapping and publishing data can be considered software engineering tasks, and the cost of fixing a bug rises exponentially when a task progresses (Boehm, 1981). In software development, a common way

to validate correct behaviour of a function is to accompany it by a set of unit tests. Similarly, a data mapping function can be accompanied by a set of test cases assigned to the mappings to ensure the correct generation of RDF datasets from input data. In this respect, incorporating quality assessment as part of the mapping and publishing workflow becomes essential, especially taking into account that it prevents the same violations to appear repeatedly within the dataset and over distinct entities. After all, in the mapping phase, structural adjustments can still be applied easily, since it allows us to pinpoint the origin of the violation, reducing the effort required to act upon quality assessment results.

Our approach has two main pillars: (i) *uniform quality assessment* of mapping definitions and the resulting dataset, as their quality is closely related; and (ii) *mapping definition refinements* to automatically improve mappings when problems are detected at the quality assessment.

Uniform quality assessment

Instead of assessing an RDF dataset for its schema quality, we apply the quality assessment to the mapping definitions directly, *before* they are used to generate the RDF dataset. Their assessment results are correlated, since MDs specify how the dataset will be formed. For example, violations of the range of a certain property can be assessed by inspecting the corresponding MD, which defines how triples with this property are generated. Even though quality assessment of MDs can cover many violations related to vocabularies and ontologies used to annotate the data, some schema-related violations depend on how the MDs are *instantiated* on the original data. For example, a violation occurs if an object of integer datatype is instantiated with a floating-point value from the original source. Therefore, a *uniform* way of incrementally assessing the quality of the RDF dataset and the mapping definitions should cover both the mappings and the dataset.

Mapping definition refinements

If violations are only corrected in the resulting dataset, they will have to be corrected every time a new version of the dataset is generated. Also, when a violation is found, it is not straightforward to discover its cause, as the connection with the MDs and the source data is not apparent. A more effective approach is to refine the MDs that generate those triples, so the violation cannot occur in future versions. Furthermore, if the violation is associated with a MD, it can be addressed directly on the place where it occurred, and instead of having to regenerate the entire dataset, only the triples affected by the refinement need to be regenerated to correct the violation.

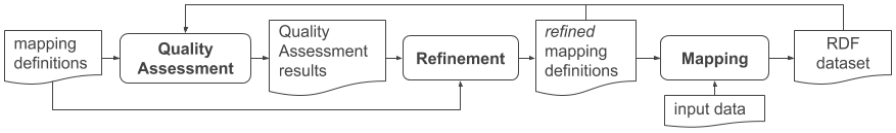


Figure 31: Quality Assessment enabled Linked Data mapping and publishing workflow

LINKED DATA AND MAPPINGS ASSESSMENT AND REFINEMENT

Uniform quality assessment requires that, in addition to the generated dataset, the mapping definitions themselves are also RDF triples. This way, the same RDF-based techniques can be applied. Additionally, performing (automated) refinement of MDs requires that machines can process and update them. Such direct processing of MDs is difficult if mappings are tightly coupled to the implementation, as is the case with most existing mapping solutions. In this respect, we focus on RDF-based mapping languages for stating the MDs. Below, we describe such a workflow (Section 12.2.1) and a solution that materializes it (Section 12.2.2).

Linked Data & Mappings Assessment & Refinement Workflow

We propose a *uniform, iterative, incremental assessment and refinement workflow* that produces, at the end, a high-quality RDF dataset. Its steps are explained below and presented in Fig. 31.

1. The schema, as stated in the MDs, is assessed against different quality assessment measures, as it would have been done if it was the actual dataset.
2. The Quality Assessment report lists each violation identified.
3. The Mapping Quality Assessment (MQA) results are used to refine the MDs. The MQA can be repeated until a set of MDs without violations is generated or if the MDs can not be further refined.
4. A refined version of the MDs is generated and used to execute the mapping of data – or a sample of the data.
5. The generated RDF output is assessed, using the same quality assessment framework. The Dataset –and optionally the Mapping– Quality Assessment (DQA) can be repeated until an ameliorated set of MDs is generated.
6. When the MDs are finalized, the actual mapping is performed and the RDF dataset is generated exempt of violations to the greatest possible extent.

Quality Assessment & Refinement with [R2]RML & RDFUnit

We provide a solution that implements the aforementioned workflow. The two main components of our solution are: the RML mapping language (Section 12.2.2.1) that uses mapping definitions expressed in RDF, a prerequisite for uniform quality assessment and automated refinements, as we discussed above, and the RDFUnit validation framework (Section 12.2.2) due to its associated test-case-based architecture (Kontokostas et al., 2014b).

RML

R2RML (Das, Sundara, and Cyganiak, 2012) is the only W3C standardised mapping language for defining mappings of data in relational databases to the RDF data model. Its extension RML (Dimou et al., 2014b) broadens its scope and covers also mappings from sources in different semi-structured formats, such as CSV, XML, and JSON. RML documents (Dimou et al., 2014b) contain rules defining how the input data will be represented in RDF. The main building blocks of RML documents are Triples Maps (Listing 12: line 1). A Triples Map defines how triples of the form (subject, predicate, object) will be generated.

A Triples Map consists of three main parts: the Logical Source, the Subject Map and zero or more Predicate-Object Maps. The Subject Map (line 2, 6) defines how unique identifiers (URIs) are generated for the resources mapped and is used as the subject of all RDF triples generated from this Triples Map. A Predicate-Object Map (line 3) consists of Predicate Maps, which define the rule that generates the triple's predicate (line 3) and Object Maps or Referencing Object Maps (line 4), which define how the triple's object is generated. The Subject Map, the Predicate Map and the Object Map are Term Maps, namely rules that generate an RDF term (an IRI, a blank node or a literal). 5A Term Map can be a *constant-valued term map* (line 3) that always generates the same RDF term, or a *reference-valued term map* (line 6) that is the data value of a referenced data fragment in a given Logical Source, or a *template-valued term map* (line 2) that is a valid string template that can contain referenced data fragments of a given Logical Source.

```
1 <#Mapping> rml:logicalSource <#InputX> ;
2   rr:subjectMap [ rr:template "http://ex.com/{ID}"; rr:class foaf:Person
      ];
3   rr:predicateObjectMap [ rr:predicate foaf:knows;
4     rr:objectMap [ rr:parentTriplesMap <#Aquihntance> ].
5 <#Aquihntance> rml:logicalSource <#InputY> ;
6   rr:subjectMap [ rml:reference "Aquintance"; rr:termType rr:IRI; rr:
      class ex:Person ] ].
```

Listing 12: RML mapping definitions

[R2]RML MAPPING DEFINITIONS QUALITY ASSESSMENT

It is straightforward to process [R2]RML mapping definitions as datasets, because they have a native RDF representation and are written from the viewpoint of the generated triples. Our assessment process targets both (i) consistency validation of the mapping definitions against the R2RML and RML schema and, mainly, (ii) consistency validation and quality assessment of the dataset to be generated against the schema defined in the mapping definitions. This first point is handled directly by RDFUnit; the second point is handled by emulating the resulting RDF dataset to assess its schema conformance.

CONSISTENCY VALIDATION OF THE MAPPING DEFINITIONS The validation of mapping definitions against the [R2]RML schema is directly handled by RDFUnit extending the supported OWL axioms. New RDFUnit TAGS were defined to support all OWL axioms in [R2]RML ontology, e. g.each Triples Map should have exactly one Subject Map, producing a total of 78 automatically generated test cases.

CONSISTENCY VALIDATION AND QUALITY ASSESSMENT OF THE DATASET AS PROJECTED BY ITS MAPPING DEFINITIONS In order to assess a dataset based only on the mapping definitions that state how it is generated, we considered the same set of schema validation patterns normally applied on the RDF dataset (cf. Table 33). Nevertheless, instead of validating the predicate against the subject and object, we extract the predicate from the Predicate Map and validate it against the Term Maps that define how the subject and object will be formed. For instance, the extracted predicate expects a Literal as object, but the Term Map that generates the object can be a Referencing Object Map that generates resources instead.

To achieve this, the properties and classes in the MDS are identified and their namespaces are used to retrieve the schemas and generate the test cases as if they were the actual dataset. We extended the corresponding RDFUnit test cases to apply to the MDS, adjusting the assessment queries.[4] For instance, the WHERE clause of the SPARQL test case that assesses a missing language is:

```
?resource ?P1 ?c .
FILTER (lang(?c) = '')
```

In order to detect the same violation directly from a mapping definition, the WHERE clause of the assessment query is adjusted as follows:

```
?poMap rr:predicate ?P1 ;
       rr:objectMap ?resource .
?P1 rdfs:range rdf:langString .
FILTER NOT EXISTS {?resource rr:language ?lang}
```

4 https://github.com/AKSW/RDFUnit/blob/master/data/tests/Manual/www.w3.org/ns/r2rml/rr.tests.Manual.ttl

The validation is *Predicate-Map-driven* in principle. The expected value (line 3), as derived from the Predicate Map, is compared to the defined one (line 4), as derived from the corresponding Object Map. The next example is an RDFUnit SPARQL test case for assessing if the `rdf:type` of a triple's ObjectMap conforms to the `rdfs:range` definition of an object property. Applying this test case to the aforementioned MD (cf. Listing 12), a violation is registered, as `foaf:knows` has `foaf:Person` and not `ex:Person` as range – assuming the ontology does not define `ex:Person` as equivalent or subclass of `foaf:Person`.

```
1  SELECT DISTINCT ?resource WHERE {
2    ?mappingTo rr:subjectMap ?resource .
3    { ?resource rr:class ?T1 . } UNION {
4      ?mapping rr:predicateObjectMap ?classPoMap .
5      ?classPoMap rr:predicate rdf:type ;
6                  rr:objectMap/rr:constant ?T1 . }
7    ?mappingFrom rr:predicateObjectMap ?poMap .
8    ?poMap rr:predicate/rdfs:range ?T2  ;
9           rr:objectMap ?objM .
10   ?objM rr:parentTriplesMap ?mappingTo .
11   FILTER NOT EXISTS {
12    ?T2 (rdfs:subClassOf|(owl:equivalentClass|^owl:equivalentClass))* ?T1
         .}}
```

In order our assessment to be complete, the defined test cases cover all possible alternative ways of defining equivalent MDs that generate the same triples. For instance, the default way to generate the type for a resource is through the rr:class property in the Subject Map (e. g.line 2 of Listing 12). However, one may also define the type via a Predicate Object Map having rdf:type in its Predicate Map.

RDFUnit can annotate test cases by requesting additional variables and binding them to specific result properties. Using the example of Listing 12.3 we map for instance, variable `?T1` as `spin:violationValue` and variable `?T2` as the expected class. When a violation is identified, the annotations are applied and a result like the following is registered:

```
1  <5b7a80b8> a rut:ExtendedTestCaseResult;
2    rut:testCase rutt:rr-produces-range-errors ;
3    # (...) Further result annotations
4    spin:violationRoot ex:objectMapX ;
5    spin:violationPath rr:class ;
6    spin:violationValue ex:Person ;
7    rut:missingValue foaf:Person ;
8    ex:erroneousPredicate  foaf:knows ;
```

However, some of the test cases normally applied to a dataset rely on the final values or refer to the complete dataset and thus, can only be validated after the mapping is performed –detected at *data-level* assessment (DQA). Such examples are (qualified) cardinality, (inverse) functionality, (a)symmetricity and irreflexivity. For example, we cannot validate an inverse functional property such as foaf:homepage with-

out the actual values. Invalid mappings can occur as the mapping definitions are instantiated based on the input source, even though the mapping definitions appear to be valid. For instance, if the input data returns, a value like "American", instead of "http://dbpedia.org/resource/United_States it would result in generating the URI `<American>`, which is invalid.

[R2]RML REFINEMENTS BASED ON QUALITY ASSESSMENT

The results of Mapping Quality Assessment (MQA) can be used to suggest modifications or even automatically refine mapping definitions. The RDFUnit ontology provides multiple result representations in different formats (Kontokostas et al., 2014b),including RDF-based serialisations (rut:ExtendedTestCaseResult result type). Therefore, its results are easily processable by an agent that can automatically add and delete triples or suggest actions to the data publisher. In Table 33, we outline all examined violation patterns and indicate which Term Map should be refined and how. The suggested refinements are the minimum required actions to be taken to refine the mapping definitions, e. g.turn an Object Map to generated resources instead of literals, and serve as indicative *proof-of-concept* of the automation's feasibility.

MAPPING REFINEMENTS. Dealing with *range-level* violations requires different actions, depending on the value of the Object Map or Referencing Object Map. The Predicate Map is used to retrieve the property and identify its range, which is then compared to the corresponding Object Map or Referencing Object Map.

If the Predicate Map contains an *object property*, for instance, but the object is generated by a Referencing Object Map, which generates resources with type different than the predicate's range –as defined by the corresponding vocabulary or ontology, the predicate's range is added as class to the Referencing Object Map. Such a violation was reported at the example mentioned in the previous section (Section 12.4). Besides manual adjustments like defining `ex:Person` as equivalent or a subclass of `foaf:Person`, the statement that the Referencing Object Map type should be a `ex:Person`, can be replaced by a `foaf:Person`:

```
DEL: ex:objectMapX rr:class ex:Person .
ADD: ex:objectMapX rr:class foaf:Person.
MOD: adjust the definition of ex:Person
```

Automatically refining *domain-level* violations requires comparing recursively the type(s) assigned to the Subject Map with each predicate's domain, as specified at the different Predicate Maps. If not explicitly defined or inferred via a subclass, the predicate's domain is additionally assigned. This also requires a follow-up check for disjoint classes, which is of crucial importance especially when composition of different vocabularies and ontologies occurs.

OWL axiom – Violation type	Level	Expect	Define	Automatic refinement
class disjointness	E	SbjMap	SbjMap	–
property disjointness	E	PreMap	PreMap	–
`rdfs:range` – class type	E	PreMap	(Ref)ObjMap	DEL: ObjMap ADD: PreMap domain to RefObjMap
`rdfs:range` – IRI instead of literal	E	PreMap	(Ref)ObjMap	DEL: (Ref)ObjMap ADD: ObjMap with literal termType
`rdfs:range` – literal instead of IRI	E	PreMap	ObjMap	DEL: ObjMap ADD: (Ref)ObjMap or ADD: ObjMap with IRI termType
`rdfs:range` – missing datatype	E	PreMap	(Ref)ObjMap	DEL: ObjMap ADD: ObjMap with PreMap datatype
`rdfs:range` – incorrect datatype	E	PreMap	(Ref)ObjMap	DEL: (Ref)ObjMap ADD: ObjMap with PreMap datatype
missing language	E	ObjMap	ObjMap	–
`rdfs:domain`	E	PreMap	SbjMap	ADD: PreMap domain to SbjMap
missing `rdf:type`	W	SbjMap	SbjMap	ADD: PreMap domain to SbjMap
deprecation	W	PreMap	PreMap	–
`owl:complementOf`	W	PreMap	SbjMap	–

Table 33: Violations detected by assessing the mapping definitions. The first column describes the type of violation, the second its level (Warning or Error). The third specifies the expected RDF term according to the ontology or schema, while the fourth the term map defining how the RDF term is generated. The last specifies the refinement.

MAPPING REFINEMENTS BASED ON DATASET QUALITY ASSESSMENT. Violations identified when the MDs are instantiated with values from the input source, can lead to a new round of refinements, if violations can be associated with a certain MD.

MAPPING REFINEMENTS IMPACT ON DATASET QUALITY. The number of automated resolutions for violations detected at the mapping level depends on (i) the number of iterations over the data chunks of the input source (e. g.number of rows), (ii) the number of references to the input source (e. g.number of referred columns) and (iii) the number of returned values from the input source for each reference. To be more precise, if $\mathcal{I}$ is the number of iterations, $\mathcal{R}$ is the set of references the input source, and $V(r)$ values are returned for $r \in R$, then the total number of errors per violation is equal to the number of triples generated from this mapping definition: $I \cdot \prod_{r \in R} V(r)$. This means that the number of errors per violation identified (and resolved) at mapping level grows linearly in function of the number of iterations, and increases drastically if multiple references and returned values exist. For instance, assuming a mapping definition with 2 references to the input, where up to 3 values can be returned

for each reference, contains a violation. Applied to an XML file with 1,000 elements, this could cause up to 6,000 error-prone triples in the worst case.

USE CASES AND ADOPTION

Our Mapping Assessment and Refinement workflow with RML and RDFUnit is already being used in multiple different contexts. The DBpedia community adapted our mapping assessment solution to improve its mappings. Other popular, medium-sized datasets also benefit of our solution to ameliorate their mappings, such as DBLP. Moreover, various projects fully relied on our solution for their dataset generation, such as CDFLG and iLastic. Last, the proposed workflow was used to refine a challenge submission. Every dataset is unique in the way mappings are applied and different types of errors arise in each case. We indicatively describe a number of representative use cases below.

DBPEDIA ((Lehmann et al., 2015)) provides a *collaborative mapping approach* of Wikipedia infoboxes to the DBpedia ontology[5] through the *DBpedia mappings wiki*[6]. DBpedia uses a wiki markup syntax for the mapping definitions and the output is adjusted in conformance to the DBpedia ontology. Although DBpedia uses the same wikitext syntax as Wikipedia –its original source– to define the MDs, the quality of wikitext-based MDs cannot be assessed directly, and thus certainly not in the same way as their resulting dataset. Thus, we automated the conversion of all DBpedia mappings to RML in order to make them processable from our tool stack. We introduced *wikitext serialisation* as a new Reference Formulation, since RML can be extended to express MDs for any type of input source. In total, we generated 674 distinct mapping documents for English, 463 for Dutch and a total of 4,468 for all languages. We used the DBpedia 2014 release and focused on a complete evaluation on the English and Dutch language editions as well as a mapping-only evaluation of all languages supported in the DBpedia mappings wiki. DBpedia originates from crowdsourced and semi-structured content and can thus be considered a noisy dataset. The MQA report was provided to the DBpedia community[7], who took advantage of it to manually refine DBpedia MDs. Automated refinements were not applicable in this case—as DBpedia framework still functions with the original MDs in wiki markup—but suggestions were provided.

5 http://wiki.dbpedia.org/Ontology
6 http://mappings.dbpedia.org
7 http://goo.gl/KcSu3E

FACETED DBLP The Computer Science bibliography (DBLP) collects open bibliographic information from major computer science journals and proceedings. Faceted DBLP builds upon the DBLP++ dataset, an enhancement of DBLP, originally stored in a mysql database. DBLP MDs are originally defined using D2RQ and were converted to RML using D2RQ-to-R2RML[8] to be processable by our workflow. DBLP, is a *medium-sized* dataset of very good quality according to our evaluation. Nonetheless, the workflow resulted in improvements.

CONTACT DETAILS OF FLEMISH LOCAL GOVERNMENTS DATASET (CDFLG)[9] In the scope of the EWI[10] project, the CDFLG dataset was generated using our workflow (De Vocht et al., 2014). This is a real case of contact details for local governments in Flanders. CDFLG is annotated using the OSLO ontology [11], defined by the Open Standards for Linking Governments Working Group (V-ICT-OR, OSLO) under the OSLO (Open Standards for Local Administrations) Programme. Two subsequent versions of its RML mapping definitions were used to generate this dataset were assessed for their quality. The decrease of mapping violations over the mapping evolution indicates that our methodology can correctly identify errors.

ILASTIC [12] Our methodology was used in a use case for iMinds[13], a research institute founded by the Flemish Government, which published its own data regarding researchers, publications, projects, external partners etc., using the proposed workflow. The mapping definitions that specify how the data is mapped to the RDF model were stated using RML. After the primary mapping definitions were stated, they were fed to our proposed implementation and were refined twice, once based on the MQA results and once based on the DQA results, leading to their final version which is free of violations.

CEUR-WS The ESWC2015 Semantic Publishing Challenge (SPC)[14] is focused on refining and enriching CEUR-WS[15] linked dataset originally generated at the ESWC2014 edition[16]. It contains Linked Data about workshops, their publications and their authors. The workflow was well aligned with the requirements of this year's challenge and was used to evaluate last year's submission based on RML (Dimou et al., 2014a) and refine it to produce the base for this year's submission (Heyvaert et al., 2015).

8 https://github.com/RMLio/D2RQ_to_R2RML.git
9 http://ewi.mmlab.be/cd/all
10 http://ewi.mmlab.be
11 http://purl.org/oslo/ns/localgov#
12 http://explore.ilastic.be/
13 http://iminds.be
14 https://github.com/ceurws/lod/wiki/SemPub2015
15 ceur-ws.org
16 http://challenges.2014.eswc-conferences.org/index.php/SemPub

	Dataset Assessment					Mapping Assessment					Affect.
Dataset	Size	TC	Time	Fail.	Viol.	Size	Time	Fail.	Viol.	Ref.	triples
DBpEn	62M	9,458	16.h	1,128	3.2M	115K	11s	1	160	–	255K
DBpNL	21M	10,491	1.5h	683	815K	53K	6s	1	124	–	106K
DBpAll	–	–	–	–	–	511K	32s	1	1,316	–	–
DBLP	12M	462	12h	7	8.1M	368	12s	2	8	6	8M
iLastic	150K	690	12s	23	37K	825	15s	3	26	23	37K
CDFLG	0.6K	2068	7s	15	678	558K	13s	4	16	13	631
CEUR-WS	2.4K	414	6s	7	783	702	5s	3	12	7	783

Table 34: Evaluation results summary. In the *Dataset Assessment* part, we provide the Size (number of triples), number of test cases, evaluation Time, Failed test cases and total individual Violations. In the *Mapping Assessment* part, we provide the mapping document Size (number of triples), evaluation Time, Failed test cases and Violation instances. Finally, we provide the number of dataset violations that can be addressed refining the mappings and estimated corresponding dataset violations that are resolved.

EVALUATION AND DISCUSSION

The datasets mentioned in Section 12.5 were used for our evaluation. In this section, the results are described and certain observations are discussed in more details for each dataset and overall. The results are aggregated in Table 34 and are available at `http://rml.io/data/ISWC15`. For our evaluation, we used an 8-core Intel i7 machine with 8GB RAM and 256 SSD HD.

Overall, it is clear that the *computational complexity and time are significantly reduced* when assessing the mapping definitions compared to the complete RDF dataset (cf. Table 34). It takes 11 seconds to assess the approximately 700 mappings of English DBpedia, compared to assessing the whole DBpedia dataset that takes of the order of several hours. In the latter case, the assessment requires examining each triple separately to identify, for instance, that 12M triples violated the range of foaf:primaryTopic, whereas with our proposed approach, only 1 triple needs to be examined. It is indisputable the workflow's *effectiveness*, as, in all cases that the dataset generation fully relies on its mapping definitions, the majority of violations is addressed. Moreover, if a set of RML mapping definitions is assessed for its quality, for every other new data source also mapped using these mapping definitions, the quality assessment does not need to be repeated for that part. Next, we discuss the results for each dataset in details:

DBPEDIA Most violations in DBpedia have *range-level* origin. When RDF is generated from the wikitext, the object type is not known and may result in wrong statements, as the DBpedia extraction framework automatically adjusts the predicate/object extraction according to the DBpedia ontology definitions. *Domain-level* violations occur as

well, because users manually provide the class a Wikipedia infobox is mapped to and the ontology properties each infobox property will use. Our framework can, in this case, identify mismatches between the user-defined class and the rdfs:domain of each provided property. We observe that 8% of the errors in DBpedia in English and 13% of DBpedia in Dutch can be fixed directly at *mapping-level*. Not only the errors as such are directly pinpointed, but it also takes negligible time to have the refinements of the violations accomplished. The evaluation of all mappings for all 27 supported language editions resulted in a total of 1316 *domain-level* violations.

DBLP dataset has 7 individual violations, leading to 8.1M violated triples. The `swrc:editor` predicate defined in a Predicate Map expects a resource of `swrc:Person` type for its `domain` instead of `foaf:Agent` as defined in the corresponding Subject Map causing 21K errors. Similarly, approximately 3M errors occurred because a Predicate Map exists with `dcterms:bibliographicCitation` as its value whose `rdfs:domain` is `bibo:BibliographicResource`. However, the corresponding Subject Map(s) generate resources of type `dcmitype:Text`, `foaf:Document` or `swrc:Book` but definitely not the expected one, thus data publishers should remain warned for potential contradictions. Moreover, the missing range of `foaf:page` and `foaf:homepage` can be fixed by refining the mapping definitions but, for links to external resources, it is common practice not to define their type. Except for 12K inverse functional violations for `foaf:homepage` that can not be addressed directly from the mapping definitions, all rest violations (98%) could be refined.

CDFLG In the first version of the CDFLG dataset, we found four violations: One caused by Predicate Object Maps that all have predicates that expect oslo:Address as their domain. However, the Subject Map is defined to be of type oslo:BasicAddress. In the same context, an incorrect range violation was identified for oslo:availableAt property. In general, violations related to Referencing Object Maps are among the most frequently encountered. Last, the object property schema:nationality was mapped as literal. The second version of CDFLG is a result of manually refining the mapping definitions according to the first mapping assessment's results. Besides the *domain level* violation, only few of the range violations remained (7%).

ILASTIC is particularly interesting because the workflow was used from the primary version of the mapping definitions, till they became free of violations. The first version was assessed and even contained R2RML schema violations, e. g. `rr:constant` had a string-valued object instead of a resource. If these mapping definitions were used, almost one fourth (25%) of its triples would be prone to errors. Although every violation was fixed after a couple of iterations assessing and refin-

ing the mapping definitions. For example, `cerif:isClassifiedBy` expects a `cerif:Classification` and not a `skos:Concept`, while `bibo:uri` expects a literal and not a resource as range. Similarly, `dcterms:issued` expects a literal and not `xsd:gYear`. A violation that occurred repeatedly was associated with the `cerif:internalidentifier` that requires a string-valued object, whereas it was associated with an Object Map that generated `xsd:positiveInteger` objects.

CEUR-WS 12 violations were identified in the dataset generated using RML for ESWC 2014 challenge and 10 out of them could already be detected at the mapping definitions. Most of them (7) were *domain-level* violations, annotating, for instance, resources of type `bibo:Volume` with properties for `bibo:Document` or for `<http://www.loc.gov/mads/rdf/v1#Address>`, e. g.for specifying the city, implying unwittingly that resources are both *Documents* and *Addresses*. The rest of the detected violations were related to contradicted datatypes, for instance, incorrectly specifying the datatype as `xsd:gYear`, while it is expected to be string. The mapping definitions for ESWC 2015 submission were produced using our workflow, were assessed and do not contain violations any more.

RELATED WORK

We summarize the state of the art of the relevant fields: data mappings to the RDF data model and LD quality assessment.

MAPPING LANGUAGES Several solutions exist to perform mappings from different data formats and serialisations to the RDF data model. In the case of data in XML format, existing XML solutions were used to define the mappings, such as XSLT, e. g.AstroGrid-D[17], or XPath, e. g.Tripliser[18], while the only mapping language defined specifically for XML to RDF mappings is X3ML[19]. In the same context, existing querying languages were also considered to describe the mappings, e. g.XSPARQL (Bischof et al., 2012) which is a language that combines XQuery and SPARQL or Tarql[20]. Due to the lack of query languages or other ways to refer to data in CSV format or spreadsheets, different mapping languages were occasionally defined, e. g.the XLWrap's mapping language (Langegger and Wöß, 2009) that converts data in spreadsheets to RDF, or the declarative OWL-centric mapping language Mapping Master's M^2 (O'Connor, Halaschek-Wiener, and Musen, 2010) that converts data from spreadsheets into the Web Ontology Language (OWL). For relational databases, different map-

17 http://www.gac-grid.de/project-products/Software/XML2RDF.html
18 http://daverog.github.io/tripliser/
19 https://github.com/delving/x3ml/blob/master/docs/x3ml-language.md
20 https://github.com/cygri/tarql

ping languages were defined (Hert, Reif, and Gall, 2011), but the W3C-standardized R2RML prevailed.

CONCLUSIONS AND FUTURE WORK

In this chapter, we propose a methodology for assessing Linked Data quality for data originally stemming from semi-structured formats. We propose a workflow that relies on assessing the mapping definitions, rather than the RDF dataset they generate. The assessment report points exactly to the root causes of the violations and can be actively used to refine the mapping definitions. The automation of refinements or suggestions is facilitated based on a comprehensive analysis of different cases, and encountered violations are addressed at the origin. This essentially allows publishers to catch and correct violations before they even occur; moreover, fixing violations early avoids propagation where one flawed mapping rule leads to many faulty triples. The evaluation shows that our methodology is applicable to i) datasets without native [R2]RML mapping definitions, such as DBLP, ii) large datasets, such as DBpedia, as well as iii) datasets in the whole process of defining their mappings, such as iLastic. It was proven that assessing the quality of mapping definitions is more efficient in terms of computational complexity, and requires significantly less time to be executed compared to assessing the entire dataset. As our evaluation indicates, it takes only a few seconds to assess the mapping definitions, while it can be time-consuming and performance-intensive when this happens at dataset level. Especially with large datasets, this can take up to several hours. Our methodology was adopted by both the community of significant public datasets, such as DBpedia, and several projects, resulting in published Linked Data of higher quality. In the future, we plan to automate and improve the application of mapping definition refinements and integrate this step into the workflow of an interactive user interface.

13

SEMANTICALLY ENHANCED QUALITY ASSURANCE: THE JURION BUSINESS USE CASE

Kontokostas, Mader, Dirschl, Eck, Leuthold, Lehmann, and Hellmann, (2016)

The publishing industry is - like many other industries - undergoing major changes. These changes are mainly based on technical developments and related habits of information consumption[1]. The world of customers has changed dramatically and as an information service provider, Wolters Kluwer wants to meet these changes with adequate solutions for customers and their work environment. For a couple of years, Wolters Kluwer has already engaged in new solutions to meet these challenges and to improve processes for generating good quality content in the backend on the one hand and to deliver information and software in the frontend that facilitates the customer's life on the other hand.

One of these frontend applications is a platform called JURION.[2] JURION is a legal information platform developed by Wolters Kluwer Germany (WKD) that merges and interlinks over one million documents of content and data from diverse sources, such as national and European legislation and court judgments, extensive internally authored content and local customer data; as well as social media and web data (e.g. from DBpedia). In collecting and managing this data, all stages of the Data Lifecycle are present – extraction, storage, authoring, interlinking, enrichment, quality analysis, repair and publication. On top of this information processing pipeline, the JURION development teams add value through applications for personalization, alerts, analysis and semantic search.

Based on the FP7 LOD2 project[3], parts of the *Linked Data stack*[4] have been deployed in JURION to handle data complexity issues. LOD2 aimed at developing novel, innovative Semantic Web technologies and also at the expansion and integration of openly accessible and interlinked data on the web. More detailed information can be found in (Auer, Bryl, and Tramp, 2014). WKD acted as a use case partner for these technologies, supported the development process of semantic technologies and integrated them to support the expansion of linked data in business environments. The software development process and data life cycle at WKD are highly independent from each other and require extensive manual management to coordinate their parallel development, leading to higher costs, quality issues and a slower time-to-market. This is why the JURION use case presented

1 For example: http://hmi.ucsd.edu/pdf/HMI_2009_ConsumerReport_Dec9_2009.pdf
2 See JURION website https://www.jurion.de/de/home/guest
3 http://lod2.eu/Welcome.html
4 http://stack.linkeddata.org/

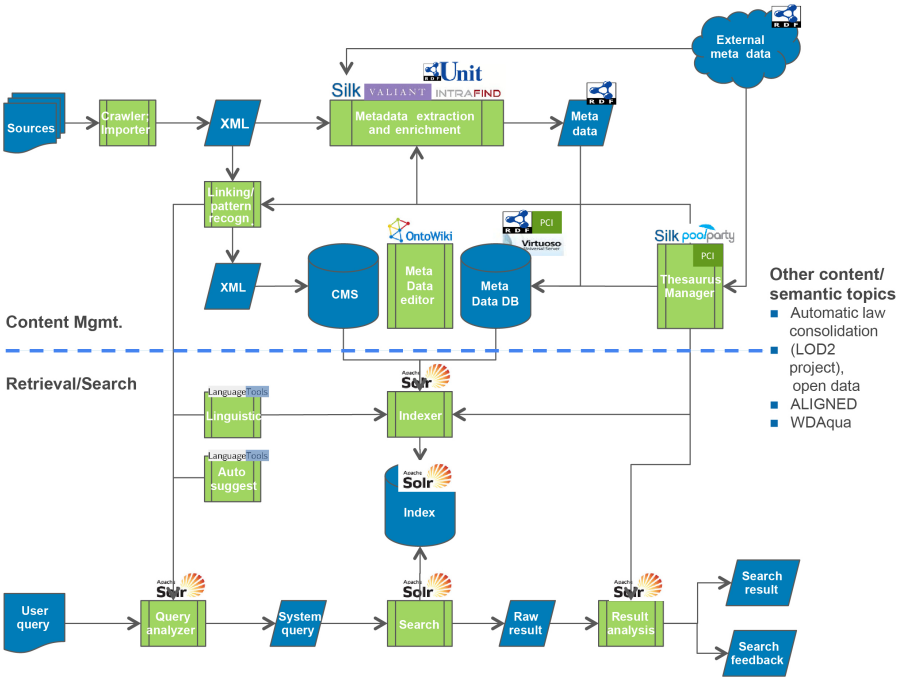


Figure 32: JURION content pipeline and semantic search

here is situated within both the Software Engineering as well as in the Data Processing area.

Through the ALIGNED project[5], JURION focuses on closing the gap between Software & Data Engineering. This chapter describes the JURION results of the first phase of the the project. In this phase, we concentrated mainly on the enhancement of data quality and repair processes. We created novel approaches for integrating RDF tools in the existing software engineering tool stack and created bindings to widely used Java libraries. We additionally created a link validation service for cleaning up external metadata residing in our databases. As a proof of concept, we open sourced some of our extensions and provide a screencast of the prototype implementation.

This chapter is structured as follows: Section 13.1 provides a detailed description of JURION and its architecture. Section 13.2 describes the challenges that drove this development. Section 13.3 provides the detailed approach we took for tackling each challenge using RDFUnit. We provide an in-depth evaluation in Section 13.4 and conclude in Section 13.5.

THE JURION BUSINESS USE CASE

WKD is a leading knowledge and information service provider in the domains of law, companies and tax and offers high quality business information for professionals. This information is more and more in-

5 http://aligned-project.eu

tegrated in digital solutions and applications. When using these solutions, customers can make critical decisions more efficiently and they can enhance their productivity in a sustainable way. Wolters Kluwer n.v. is based in more than 40 countries and serves customers in more than 175 countries worldwide.

JURION is the legal knowledge platform developed by WKD. It is not only a legal search platform, but considers search for legal information as an integrated part of the lawyer's daily processes. JURION combines competencies in the areas of legal publishing, software, portal technology and services, which cover all core processes of the lawyer within one single environment by connecting and integrating many different internal and external data sources.

The main goal of JURION is not to be yet another search engine. On the one hand, because Google as a reference application has made major progress in recent times, even in search environments dedicated to professionals. On the other hand, legal research is just one part of the lawyer's main and daily tasks. So the higher the coverage of core processes of a digital offering, the more added-value on the customer's side is generated and the higher the willingness to pay for that service will be. In addition, the more touchpoints between vendor and customer exist, the lower is the possibility for the service provider to be replaced by others.

Figure 32 describes the overall JURION content processing and retrieval infrastructure. Within the content pipeline, metadata is extracted from the proprietary WKD XML schema and transformed in RDF. In the thesaurus manager, controlled vocabularies and domain models based on SKOS standard are created, maintained and delivered for further usage. The indexing process of a search engine includes more and more additional information on top of the pure text. Queries are analyzed for legal references and keywords, which are matched against existing data in the metadata management systems. Once there are matches, the semantic relations are shown in the results overview by specific references to texts and related knowledge visualizations.

Since most of these new service offerings like workflow support at the lawyer's desk are not document and content driven, the current paradigm of using pure XML as the only major data format had to be given up. Data and metadata are driving internal processes and therefore most of the features and functionalities in the JURION application. So, this data must not be locked in DTDs or XML schemas anymore. Conversion of this data in traditional RDBMS would have been possible, but the main benefits of these systems like high performance and robustness were not the major requirements in this setting. Instead, the data needed to be stored and maintained in a much more flexible and 'inter-connectable' format, so that new data format requirements like adding new metadata or a new relationship

type could be processed in a more or less real-time fashion. Semantic web technologies were chosen to meet that need, since e.g. their triple store technology supports this flexibility and since high performance is as already mentioned not a major requirement in a CMS environment. In addition, due to government initiatives on a national and European level, quite a lot of valuable data in the legal field was available in SKOS and RDF format, so that the integration effort for this data was rather limited, as soon as the basic technical infrastructure was laid. Once the internal and integrated external data was available, new features like personalization based on domain preferences (e.g. boosting labor law documents in the result list, based on current search behavior), context-sensitive disambiguation dialogues to resolve query issues (e.g. 'contract' as 'labor contract' or 'sales contract') or the sustainable linking to external data sources like EU directives and court decisions with their own legal contexts, e.g. across languages and countries could be established.

Related Work

In industrial settings, architectural system details are most times kept hidden due to conflicts of interest. However, major companies in the media & publishing sector are using semantic web technologies. In 2013, BBC published their internal knowledge graph[6]. BBC keeps an open position on Linked Data and publishes many of their semantic-web-enabled features[7]. Thomson Reuters provides B2B semantic solutions with OpenCalais[8] and PermID[9]. Guardian uses Linked Data to augment the data they provide behind a paid API[10]. At the end of 2015, Springer announced a semantic wrapper of their data[11]. Nature Publishing Group has been an early adopter on linked data[12]. Finally, Pearson is also observed to use semantic web technologies[13].

CHALLENGES

JURION merges and interlinks over one million documents of content and data from diverse sources. Currently, the software development process and data life cycle are highly independent from each other and require extensive manual management to coordinate their parallel development, leading to higher costs, quality issues and a slower time-to-market. The higher flexibility of the data model as well as the

6 http://www.bbc.co.uk/things/
7 http://www.bbc.co.uk/blogs/internet/tags/linked-data
8 http://www.opencalais.com/
9 https://permid.org/
10 http://www.theguardian.com/open-platform/blog/linked-data-open-platform
11 http://lod.springer.com/wiki/bin/view/Linked+Open+Data/About
12 http://www.nature.com/ontologies/
13 https://www.semantic-web.at/pearson

shortened time-to-market for data features can only be materialized when most of the testing and QA effort – after data & schema changes are introduced – are tackled in a semi-automatic fashion. Thus, benefits like flexibility and scalability only materialize when it is not necessary to do all the quality checks manually, or involving expensive domain experts.

As depicted in Figure 32, within the content pipeline, metadata is extracted from the proprietary WKD XML schema and transformed in RDF. Due to regular changes in the XML format, the correct transformation process based on existing XSLT scripts must be secured, so that no inconsistent data is fueled into the metadata database. In the thesaurus manager, controlled vocabularies and domain models based on SKOS standard are created, maintained and delivered for further usage. The integrity of the knowledge management system as a whole needs to be ensured. Therefore, regular local and global quality checks need to be executed, so that e.g. inconsistencies across different controlled vocabularies can be detected and resumed.

Through the ALIGNED project, we target to enable JURION to address more complex business requirements that rely on tighter coupling of software and data. In this chapter, we focus on improving the metadata extraction process as well as inconsistencies across controlled vocabularies and in particular external links coming from the JURION enrichment phase.

Metadata RDF Conversion Verification

At the top of Figure 32 of the content pipeline, metadata is extracted from the proprietary WKD XML schema and transformed to RDF. Due to regular changes in the XML format, the correct transformation process based on existing XSLT scripts must be secured, so that no inconsistent data is fuelled into the metadata database. The main challenge of this task is to reduce the error in data transformation and accelerate the delivery of metadata information to JURION.

APPROACH & GOALS Based on the schema, test cases should automatically be created, which are run on a regular basis against the data that needs to be transformed. The errors detected will lead to refinements and changes of the XSLT scripts and sometimes also to schema changes, which impose again new automatically created test cases. This approach provides:

1. better control over RDF metadata
2. streamlined transformation process from XML to RDF
3. early detection of errors in RDF metadata, since the resulting RDF metadata are a core ingredient for many subsequent process steps in production and application usage, and

4. more flexibility in RDF metadata creation

IMPACT Continuous high quality triplification of semi-structured data is a common problem in the information industry, since schema changes and enhancements are routine tasks, but ensuring data quality is still very often purely manual effort. So any automation will support a lot of real-life use cases in different domains.

Existing Infrastructure

As part of the core CMS tasks within JURION each WKD XML document is checked-in through internal workflow functionality and gets converted to RDF which is based on the "Platform Content Interface" (PCI) ontology. The PCI ontology is a proprietary schema that describes legal documents and metadata in OWL. Due to change requests and new use cases for the RDF metadata in the ontology, the conversion logic or both the conversion logic and ontology need amendments. In these cases we need to make sure that the RDF data that is generated from the WKD XML documents still complies with the PCI ontology for quality assurance.

QUALITY ASSURANCE As a gatekeeper to avoid loading flawed data into the triple store, each result of the conversion from WKD XML into PCI RDF was sent to a dedicated, proprietary validation service that inspects the input and verifies compliance with the ontology. This approach assured that the conversion results are verified but came with major issues which makes it unsuitable for ad-hoc testing and quick feedback. The three most important ones are:

- the current service only processes entire PCI-packages, i.e. several datasets; this makes error detection on single data units quite difficult and service errors block the whole processing pipeline
- the service is a SOAP based web service that operates asynchronously with many independent process steps, which imposes high complexity on its usage
- it depends on other services and requires permanent network access and therefore is potentially unstable

To improve these issues, we want to implement unit-test scenarios that can be run directly coupled to the development environment of the conversion projects and is therefore seamlessly integrated into the workflow. The tests should be run both automatically on every change in the project, but also be able to be manually triggered. Tests should be easily extendable and expressive enough to effectively spot issues in the conversion process. The feedback loop should be coupled as tightly as possible to the submitted change.

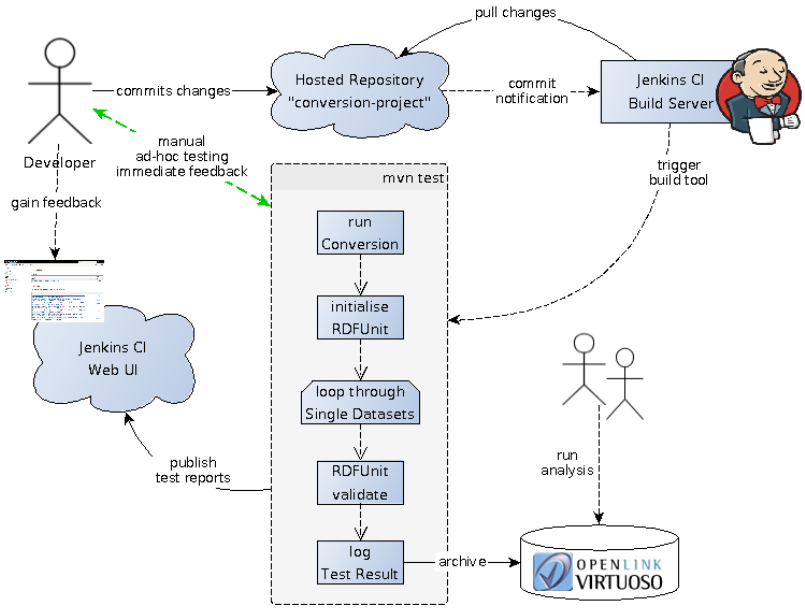


Figure 33: RDFUnit integration in JURION

SEMANTICALLY ENHANCED QUALITY ASSURANCE

The JURION challenges described in Section 13.2 are targeted by the following extensions to the JURION workflow and its components: a) verification of correct metadata conversion, b) integration of repeatable tests into the development environment and c) automation of external link clean-up. In the following sections we present our approach for tackling each challenge.

A six-minute screencast video has been developed which showcases the demonstrator described in this chapter. The screencast shows some background context on the JURION use case and the prototype implementation in action. It is available on YouTube[14] and is linked through the ALIGNED project website[15]. RDFUnit (Kontokostas et al., 2014a), including the extensions developed for this demonstrator is available as open source code. PoolParty is a commercial closed-source product[16] and the extensions developed for this demonstrator will be folded into future releases of the product when this is commercially viable.

XML to RDF Conversion Verification

To allow comparable and reproducible tests with short execution times, a number of WKD XML reference documents have been selected, against which the actual conversion into PCI RDF is executed. Each resulting RDF dataset is then verified individually. The prototyped

14 https://youtu.be/6aLXK7N7wFE
15 http://aligned-project.eu/jurion-demonstration-system/
16 https://www.poolparty.biz/

execution	source	errors	failed	testsRun	testsSuccessful	timeout	totalViolations	start	end
http://rdfunit.aksw.org/data/results#c5bfc3ba-2ae4-11b2-809b-d4bedf9680f44	rdfunit:007d1d91-30c0-4ea3-89b8-80591bdb1c30	0	27	2922	2895	0	244	2015-07-20T12:25:08.749Z	2015-07-20T12:2
http://rdfunit.aksw.org/data/results#a8f02216-2af0-11b2-808d-0050563b0112	rdfunit:009d8713-0bd5-4016-a187-a02838b4c90d	0	26	2922	2896	0	236	2015-09-17T14:38:45.202Z	2015-09-17T14:3
http://rdfunit.aksw.org/data/results#59c21393-2ae5-11b2-8022-843a4b6a105c	rdfunit:00afeffa-965c-47c9-a9a1-b047ed136dcd	0	34	2922	2888	0	303	2015-07-23T09:23:41.883Z	2015-07-23T09:2
http://rdfunit.aksw.org/data/results#2bda274a-2ae4-11b2-80cc-d4bedf9680f44	rdfunit:00d1dcd6-5b65-4af0-a26e-c077d02976ca	0	11	2922	2911	0	234	2015-07-17T12:41:54.213Z	2015-07-17T12:4
http://rdfunit.aksw.org/data/results#a97f49d9-2af0-11b2-80a1-0050563b0112	rdfunit:00ece7c5-8618-46c1-a70d-c40e493b7648	0	8	2922	2914	0	218	2015-09-17T14:54:29.594Z	2015-09-17T14:5

Figure 34: Sample of structured metadata stored in a triple store for every project build

solution (cf. Figure 33) integrates RDFUnit as the core driver of the tests. It is set upon the JUnit-API[17] so that it integrates seamlessly into the development tool chain and the software build process in general. For instance, a developer can trigger the test chain manually on his local workstation to retrieve direct feedback at any time and any change in the conversion project automatically leads to full test run which is performed by the build system.

All executed tests are based on RDFUnit's auto-generators, which derive test cases at runtime from the latest version of the PCI-ontology. As a proof of concept RDFUnit's test results (the validation model based on the Test Driven Data Validation Ontology (Kontokostas et al., 2014b)) linked to this test is stored into a Virtuoso triplestore to enable future analysis/reviews of historical data. Figure 34 shows a sample of the test results, which can be stored on every build and used on regular basis in current and future QA reports.

Early and quick feedback on changes to the project is very valuable to assure that the project is in good health and existing functionality meets the defined expectations. Good coverage with automated tests prevents bugs from slipping in released functionality which may have side effects on other parts of the system. RDFUnit enables possibilities but still needs a tighter integration as a library with our existing toolchain to improve reporting capabilities and make its feedback even more useful. RDFUnit proves as very useful and will be a fixed component of the operational tech stack within WKD JURION from now on. We will provide further requirements to improve RDFUnit's integration into our development pipeline. At a later point in time, we will utilize RDFUnit to enable monitoring the existing data store to implement quality assurance on operational side.

We additionally integrated RDFUnit with JUnit. JUnit is a unit testing framework for Java, supported by most Java developer IDEs and build tools (e.g. maven, Eclipse). JUnit allows to execute repeatable tests as part of the development workflow in line with the test-driven development paradigm. As part of the described use cases we contributed the integration of the RDFUnit-JUnit-module[18]. We added specific Java annotations on JUnit classes that can define the input dataset and the schema that RDFUnit can test (cf. Listing 13). Each test generated by RDFUnit TAGs is translated in a separate JUnit test

17 http://junit.org/javadoc/latest/

18 https://github.com/AKSW/RDFUnit/tree/master/rdfunit-junit

and reported by JUnit. This approach facilitates simpler setups and can verify specific input files. The benefit is the immediate integration of RDF dataset testing on existing software development tool stacks.

```
@RunWith(RdfUnitJunitRunner.class)
@Schema(uri = "ontologies/foaf.rdf")
public static class TestRunner {

  @TestInput
  public RDFReader getInputData() {
    return new RDFModelReader(ModelFactory
      .createDefaultModel()
      .read("inputmodels/foaf.rdf")); } }
```

Listing 13: Example JUnit definition for testing an input dataset against a schema

EVALUATION

Our methodology for baseline data collection is divided into three categories: productivity, quality and agility. The analysis is based on measured metrics and the qualitative feedback of experts and users. Participants of the evaluation study were selected from WKD staff in the fields of software development and data development. There were seven participants in total: four involved in the expert evaluation and three content experts involved in the usability/interview evaluation.

Productivity

COLLECTION METHODS & METRICS We collected content expert evaluations for the metadata extraction verification, a test suite was set up to measure metrics of productivity for all implemented features and finally, interviews were conducted to obtain feedback from prospective users of PoolParty functionalities. For the RDFUnit integration, we measured the total time for quality checks and error detection, as well as the need for manual interaction. For the external link validation we measured the number of checked links, the number of violations and the total time.

RDFUnit enabled us to develop automated tests that provide tight feedback and good integration into the existing toolchain. It enabled error messages, which point exactly to the offending resource, making bug fixing much easier. Depending on the size of the document and size of the ontology, total time to execute a test-suite varies, but can be indicated with 1ms to 50ms per single test. With this approach, the feedback is as close to real-time as possible; currently a couple of minutes. Since it is possible to trigger the quality checks manually at any time through the existing developers IDE menus, speedy performance is desirable to avoid developer idle time. Though quality

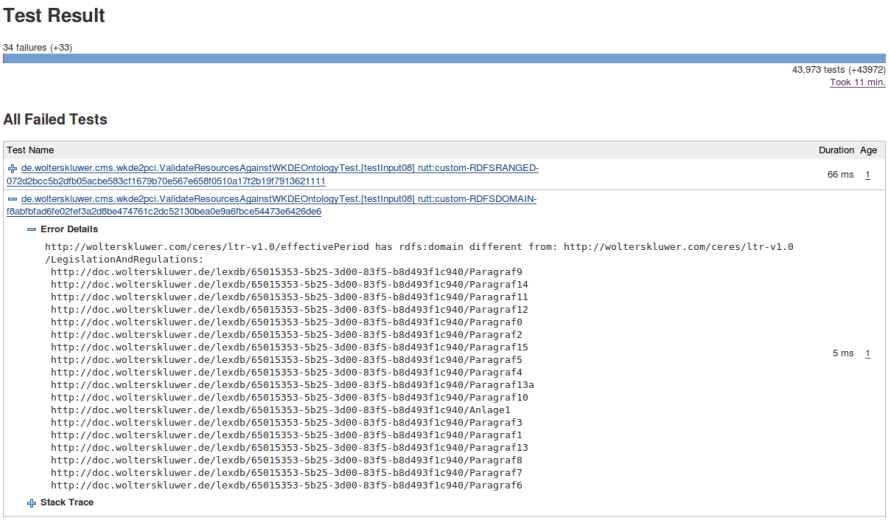


Figure 35: Jenkins Test Report

checks can be triggered by manual execution, they are always verified automatically by the build system, which sends a notification if an error occurs. Due to the development process it is guaranteed that with every change the whole set of quality checks is executed and reported automatically. The current setup generates and runs about 44000 tests with a total duration of 11 minutes which may scale-up easily when parallelized or clustered. The details-section reports each violated RDFUnit test individually with it's corresponding error message and a list of failing resources.

Quality

COLLECTION METHODS & METRICS We set up a test suite to measure quality metrics for all features, we additionally used content expert dataset evaluations. For the metadata extraction verification we collected the number of detected error categories, the test coverage and expert evaluation. For the external link validation we measured the correctness of results and usability aspects.

In the metadata extraction verification we had the following questions: "What kind of errors can be detected" and "is categorization possible"? We used the RDFUnit supported axioms to categorize the errors wherever possible. RDFUnit supports many RDFS & OWL axioms. Regarding test coverage, RDFUnit provides test coverage metrics. However, we did not yet integrate the test coverage metrics in our operational tool stack. This is a next (crucial) step, as we need to evaluate the relevance of individual tests to the tested dataset. Ideally, we would need to get the percentage of the input dataset that is covered by tests and how many of these tests actually measured features of the input dataset. From our expert evaluation we concluded that it is helpful to spot errors introduced by changes, since issues spot-

ted in this way can be assumed to point to really existing errors; the causes of which can be identified and addressed. In contrast, successful tests are less significant as we are not yet able to evaluate whether and how the measurements taken correspond to target measures and these tests do not point to concrete errors. To resolve this, we will proceed to integrate measures that help evaluating the test cases on the one hand, and the input datasets on the other hand.

Agility

COLLECTION METHODS & METRICS We collected evaluations of content experts for the RDFUnit integration and evaluations of technical experts for the PoolParty tool. We collected metrics for time to include new requirements in RDFUnit integration and for the external links validation scope of external link checks, possibility of integration, configuration time and extension.

With respect to the XML-to-PCI conversion verification, including new constraints or adapting existing constraints is a convenient process. The procedure works by adding new reference documents to the input dataset to make the test environment as representative as possible. As the process of generating tests and testing is fully automated, it adapts very easily to changed parameters. However, adding more documents to the input dataset increases the total runtime of the test-suite, which affects the time to feedback. Therefore, one must be careful with the selection of proper reference documents.

Analysis

The evaluation of the prototype shows clearly that during the first phase of prototype development, we have achieved our aim to improve the productivity and quality of data processes within the data lifecycle. With the presented features, these improvements could be shown. Performance and quality/error rates of the test results were reasonable. In addition to the tests, there will be new scope for data repair processes to correct the detected errors of the dataset, as we gained new insights into data violations (e.g. more categories of violated external links than we expected). Nonetheless, there need to be further improvements, especially with regard to usability, performance, integration of functionalities and required details that are not yet fully working.

In summary, the productivity of data processes is clearly improved by the initial prototype. The statistics and external link validation functionalities can help to save much time by replacing time consuming manual work by efficient data overviews.

Concerning the quality of the prototype functionalities, the results are very satisfying. For notifications and external link validation there

are only few issues. For the data transformation with RDFUnit there needs to be further investigations to enable comprehensive and extensive data testing results. Usability issues need to be tackled in all of the features for a better operational implementation. As this is only an initial prototype, usability was less of a focus.

The testers' feedback for agility of features is quite positive. The agility of RDFUnit is seen as satisfying, as the automated service allows the implementation of new requirements easily. External link validation has a reasonable agility and is planned to be done by an external application to address performance issues.

CONCLUSIONS & FUTURE WORK

In this chapter, we described an industrial use case of RDF technologies with JURION. We managed to weaken the gap between software and data development by integrating quality checks in our existing software tool stack. We provided a screencast of our prototype and contributed bindings between RDFUnit and JUnit as open source.

In future work we plan to improve the RDFUnit integrations in JURION. Further research is needed for test coverage reports as well as the generation test analytics. For example, time to fix a bug, identification of regressions, etc.

Part V

CONCLUSIONS

14 CONCLUSIONS AND FUTURE WORK

This chapter concludes this thesis and discusses candidate future work.

CONCLUSIONS

This thesis forms a comprehensive set of research and engineering tasks for increasing both *precision* and *recall* in *large-scale multilingual knowledge extraction*, with a focus on DBpedia. The results of this thesis are already contributed back to the scientific & industry community through an improved DBpedia open data stack and open source tools, services and specifications.

Chapter 3 provided an overview of the DBpedia project and the architecture of the *DBpedia Information Extraction Framework*. It reports the status of the active ontology and mapping community and lists three projects related to this thesis, DBTax, DataID and DBpedia Viewer. DBTax automatically generated 1.9K T-Box and 10.7M A-Box statements, DataID provides an RDF vocabulary for describing a DBpedia release as well as any RDF dataset and, DBpedia Viewer is a Linked Data interface that integrates existing DBpedia services as well as external Linked Data visualization tools to improve the visualization and exploration of DBpedia.

In Chapter 4 we described the performed extensions on DBpedia to improve and extend internationalization and localization (I18n) support. This work resulted in the increase of up to 85% in facts extracted from non-English Wikipedia language editions and bootstrapped the institutionalization of the Internationalization Committee. In addition, this was the first step towards Linked Data internationalization and the first successful attempt to serve Linked Data with de-referencable IRIs that also serves as a guide for LOD publishing in non-Latin languages. A derived work is described in (Gayo, Kontokostas, and Auer, 2013) and provides a set of predefined patters for publishing multilingual linked data.

Chapter 5 presents DBpedia Commons, the first large-scale knowledge extraction from Wikimedia Commons that resulted in 1.4 billion RDF triples on file metadata, provenance, descriptions, and license information. . Chapter 6 describes an alternative RDF representation of Wikidata. This work involved the creation of 10 new DBpedia extractors, a Wikidata-to-DBpedia mapping language and additional post-processing & validation steps. With the current mapping status we managed to generate over 1.4 billion RDF triples with CC0 license.

According to the web server statistics, the daily number of DBw visitors range from 300 to 2,700 and we counted almost 30,000 unique IPs since the start of the project.

From Chapter 8 on we listed our work related to quality assessment. This work was originally purposed to assess the quality of DBpedia alone. However, the methodology and the tooling that was developed was easily applied on many different domains and settings.

Chapter 8 and Chapter 9 provided and evaluated the *Test-Driven Quality Assessment Methodology* (TDQAM). TDQAM consists of core term definitions, a workflow for the elicitation of test cases, an extensive pattern library and an RDF vocabulary for expressing TDQAM in RDF. One major contribution from TDQAM is RDFUnit, a general-purpose & scalable quality assessment tool that implements the methodology. Using RDFUnit we managed to automatically generate 32K reusable test case for 297 RDF vocabularies and evaluated the quality of five big LOD datasets by reusing these test cases. This assessment easily and efficiently revealed millions of data quality issues in those datasets. In addition, we devised 277 test cases for NLP datasets using the *Lemon* and *NIF* vocabularies. We run these test cases on 11 datasets using those vocabularies and containing approximately 23 million triples and identified many millions of errors. We showed progress in implementing domain-specific validation by quickly improving existing validation provided by ontology maintainers

Chapter 10 described the first comprehensive empirical quality analysis for more than 500 resources of a large Linked Data dataset extracted from crowdsourced content. We found that a substantial number of problems exist in DBpedia and the overall quality, with a 11.93% error rate, is moderate. In addition to the quality analysis of DBpedia, we devised a generic methodology for Linked Data quality analysis, derived a comprehensive taxonomy of extraction quality problems and developed a tool, called TripleCheckMate, which can assist domain experts with the evaluation. This work was conducted prior to TDQAM and revealed a lot of unknown quality issues in DBpedia. However, one main inherent problem was that it relied on manual evaluation and could only evaluate data samples and not the complete dataset. In addition, we observed a very low interrater agreement that confirms that, even for domain experts, data quality can be subjective.

The main contribution of Chapter 12 is a quality assessment workflow – assisted by software – that promotes the validation of the mapping definitions before the validation of the RDF dataset that these mappings generate. An incorrect mapping definition may lead to even millions of instance data violations and our workflow allows publishers to catch and correct these violations before they even occur. We show that assessing the quality of mapping definitions is more efficient in terms of computational complexity. As our evalu-

ation indicates, it takes only a few seconds to assess the mapping definitions, while it can be very time-consuming and performance-intensive when this happens at dataset level. Especially with large datasets, like DBpedia, this can take up to several hours.

Chapter 13 showcased how RDFUnit and the *Test-Driven Quality Assessment Methodology* is successfully applied and evaluated in the production system of Wolters Kluwers Germany, a leading knowledge and information service provider in the domains of law, companies and tax. We present how we integrate RDFUnit in a continuous integration (CI) system as well as JUnit[1] to facilitate validation scenarios within the Wolters Kluwers data management workflows. This is a very important contribution of this thesis as it presents a direct use of this work from industry and the integration of RDFUnit with popular software engineering tools.

In the context of data quality, the author co-organized three workshop proceedings on Linked Data Quality (LDQ) (Debattista et al., 2016; Knuth, Kontokostas, and Sack, 2014; Rula et al., 2015) and is in the process of editing a special issue on Semantic Web Journal for *Special Issue on Quality Management of Semantic Web Assets (Data, Services and Systems)* [2]. Finally, another major impact of this thesis was the influence and authorship of SHACL, a language for defining constraints on RDF graphs. SHACL is a close to become a W3C recommendation and fill the existing validation gap in the RDF technology stack.

FUTURE WORK

Lehmann, Isele, Jakob, Jentzsch, Kontokostas, Mendes, Hellmann, Morsey, Kleef, Auer, and Bizer, (2015)

DBpedia can be seen as a proof-of-concept and blueprint for the feasibility of large-scale knowledge extraction from crowd-sourced content repositories. There are a large number of further crowd-sourced content repositories and DBpedia already had an impact on their structured data publishing and interlinking. Besides Wikimedia Commons (Chapter 5) and Wikidata (Chapter 6), two examples are Wiktionary with the Wiktionary extraction (Hellmann, Brekle, and Auer, 2012) meanwhile becoming part of DBpedia and *LinkedGeoData* (Stadler et al., 2012), which aims to implement similar data extraction, publishing and linking strategies for *OpenStreetMaps*. Nevertheless, there are many ways in which the project could be further advanced in the future:

Multilingual data integration and fusion. An area, which is still largely unexplored is the integration and fusion between different DBpedia language editions. Non-English DBpedia editions comprise a better and different coverage of local culture. When we are able to precisely identify equivalent, overlapping and complementary parts in differ-

1 JUnit: Java unit-testing framework http://junit.org
2 http://www.semantic-web-journal.net/blog/call-papers-special-issue-quality-management-semantic-web-assets-da

ent DBpedia language editions, we can reach significantly increased coverage. On the other hand, comparing the values of a specific property between different language editions will help us to spot extraction errors as well as wrong or outdated information in Wikipedia. The wikidata extraction that was discussed in Chapter 6 will play a central role in this direction.

Community-driven data quality improvement. In the future, we also aim to engage a larger community of DBpedia users in feedback loops, which help us to identify data quality problems and corresponding deficiencies of the DBpedia extraction framework. By constantly monitoring the data quality and integrating improvements into the mappings to the DBpedia ontology as well as fixes into the extraction framework, we aim to demonstrate that the Wikipedia community is not only capable of creating the largest encyclopedia, but also the most comprehensive and structured knowledge base. we were making a first step in this direction.

Feedback for Wikipedia. A promising prospect is that DBpedia can help to identify misrepresentations, errors and inconsistencies in Wikipedia. In the future, we plan to provide more feedback to the Wikipedia community about the quality of Wikipedia. This can, for instance, be achieved in the form of sanity checks, which are implemented as SPARQL queries on the DBpedia Live endpoint, which identify data quality issues and are executed in certain intervals. For example, a query could check that the birthday of a person must always be before the death day or spot outliers that differ significantly from the range of the majority of the other values. In case a Wikipedia editor makes a mistake or typo when adding such information to a page, this could be automatically identified and provided as feedback to Wikipedians. In Chapter 4 we proved that DBpedia can indeed serve as an important statistical diagnostic tool for Wikipedia that helps to identify and resolve existing and emerging issues. However, this was only a small step towards a bigger research agenda.

Integrate DBpedia and NLP. There is a big potential for employing Linked Data background knowledge in various Natural Language Processing (NLP) tasks. One very promising research avenue in this regard is to employ DBpedia as structured background knowledge for named entity recognition and disambiguation. Currently, most approaches use statistical information such as co-occurrence for named entity disambiguation. However, co-occurrence is not always easy to determine (depends on training data) and update (requires recomputation). With DBpedia and in particular DBpedia Live, we have comprehensive and evolving background knowledge comprising information on the relationship between a large number of real-world entities. Consequently, we can employ this information for deciding to what entity a certain surface form should be mapped. A step towards integrating DBpedia and NLP was the 1st *NLP and DBpedia*

Workshop that was co-co-organized by the author (Hellmann et al., 2013b). The *NLP and DBpedia workshop* is running on a yearly basis even since.

DBpedia Interlinking improvement. Another area of research is the more efficient utilization of the Wikipedia interlanguage links. The approach discussed in Section 4.3.1 was safe and straightforward. A further analysis of the *conflict situations* and how they could be resolved will be of great importance both for Wikipedia and the internationalization of the Semantic Web. The *conflict situations* analysis could also provide new data and make us re-examine the use of `owl:sameAs` – as a too strong semantic implication – with other vocabularies (i.e. SKOS). We could also utilize the *conflicts*, which are now discarded, by adding `rdfs:seeAlso` links. In addition, the management of the external DBpedia links is also an area that is not thoroughly explored.

Wikidata integration improvements. With regard to Wikidata we plan to extend the mapping coverage as well as extend the language with new mapping functions and more advanced mapping definitions. The dataset is already part of the bi-yearly DBpedia release cycle and thus regularly updated. We will additionally consider providing DBw as a live service similar to DBpedia Live.

With regard to *Test-Driven Quality Assessment Methodology* (TDQAM), we see this work as the first step in a larger research and development agenda to position test-driven data engineering similar to test-driven software engineering. As a result, we hope that test-driven data quality can contribute to solve one of the most pressing problems of the Data Web – the improvement of data quality and the increase of Linked Data fitness for use. An area that is not explored by this thesis is the generation of automatic repair strategies, i.e. use of templates and bindings to fix problems efficiently.

In the context of crowdsourcing the quality assessment, Chapter 10 reports on results by domain experts. A possible promising approach would be the combination of crowdsourcing by both domain experts and simple workers, in combination of automated approaches like TDQAM. A positive step in this direction is shown in (Acosta et al., 2013).

For mapping validation (Chapter 12), we plan to automate and improve the application of mapping definition refinements and integrate this step into the workflow of an interactive user interface.

With regard to the RDFUnit integration in the JURION platform, further research is needed for test coverage reports as well as the generation test analytics. For example, time to fix a bug, identification of regressions, etc.

BIBLIOGRAPHY

Acosta, Maribel, Amrapali Zaveri, Elena Simperl, Dimitris Kontokostas, Sören Auer, and Jens Lehmann (2013). "Crowdsourcing Linked Data quality assessment." In: *12th International Semantic Web Conference, 21-25 October 2013, Sydney, Australia*, pp. 260–276. URL: http://svn.aksw.org/papers/2013/ISWC_Crowdsourcing/public.pdf (cit. on pp. xii, 146, 185).

Adida, Ben, Mark Birbeck, Shane McCarron, and Steven Pemberton (2008). *RDFa in XHTML: Syntax and Processing – A Collection of Attributes and Processing Rules for Extending XHTML to Support RDF*. W3C Recommendation. http://www.w3.org/TR/rdfa-syntax/. URL: http://www.w3.org/TR/rdfa-syntax/ (cit. on p. 52).

Alexander, Keith, Richard Cyganiak, Michael Hausenblas, and Jun Zhao. *Describing Linked Datasets with the VoID Vocabulary*. W3C Interest Group Note. URL: https://www.w3.org/TR/void/ (cit. on p. 32).

Angles, Renzo and Claudio Gutierrez (2008). "The expressive power of SPARQL." In: *The Semantic Web-ISWC 2008*. Springer, pp. 114–129 (cit. on p. 98).

Aprosio, Alessio Palmero, Claudio Giuliano, and Alberto Lavelli (2013). "Towards an Automatic Creation of Localized Versions of DBpedia." In: *The Semantic Web–ISWC 2013*. Springer, pp. 494–509 (cit. on p. 34).

Auer, Sören, Volha Bryl, and Sebastian Tramp, eds. (2014). *Linked Open Data – Creating Knowledge Out of Interlinked Data*. 1st. Vol. 8661. LNCS VII. 215. Springer. ISBN: 978-3-319-09846-3. DOI: 10.1007/978-3-319-09846-3 (cit. on p. 167).

Auer, Sören and Dimitris Kontokostas (2014). "Towards Web Intelligence Through the Crowdsourcing of Semantics." In: *Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web Companion*. WWW Companion '14. Seoul, Korea, pp. 991–992. ISBN: 978-1-4503-2745-9. DOI: 10.1145/2567948.2578841. URL: http://dx.doi.org/10.1145/2567948.2578841 (cit. on pp. xii, 129).

Auer, Sören and Jens Lehmann (2007). "What Have Innsbruck and Leipzig in Common? Extracting Semantics from Wiki Content." In: *Proceedings of the ESWC (2007)*. Vol. 4519. Lecture Notes in Computer Science. Berlin / Heidelberg: Springer, pp. 503–517. ISBN: 978-3-540-72666-1. URL: http://jens-lehmann.org/files/2007/wiki_extraction.pdf (cit. on pp. 33, 146).

Auer, Sören, Chris Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives (2008). "DBpedia: A Nucleus for a

Web of Open Data." In: *Proceedings of the 6th International Semantic Web Conference (ISWC)*. Vol. 4825. Lecture Notes in Computer Science. Springer, pp. 722–735. DOI: doi:10.1007/978-3-540-76298-0_52 (cit. on p. 33).

Auer, Sören, Matthias Weidl, Jens Lehmann, Amrapali J. Zaveri, and Key-Sun Choi (2010). "I18n of Semantic Web Applications." In: *Proceedings of the 9th International Semantic Web Conference (ISWC2010)*. Lecture Notes in Computer Science. Berlin / Heidelberg: Springer. DOI: 10.1007/978-3-642-17749-1_1. URL: http://svn.aksw.org/papers/2010/ISWC_I18n/public.pdf (cit. on pp. 43, 50, 53).

Beckett, D. (2007). *Turtle - Terse RDF Triple Language*. Tech. rep. URL: http://www.dajobe.org/2004/01/turtle/ (cit. on p. 53).

Belhajjame, Khalid, James Cheney, David Corsar, Daniel Garijo, Stian Soiland-Reyes, Stephan Zednik, and Jun Zhao (2013). *PROV-O: The PROV Ontology*. Tech. rep. URL: http://www.w3.org/TR/prov-o/ (visited on 04/30/2013) (cit. on p. 98).

Berners-Lee, T., R. Fielding, and L. Masinter (2005). *RFC 3986, Uniform Resource Identifier (URI): Generic Syntax*. Ed. by Internet Engineering Task Force (IETF). Request For Comments (RFC). URL: http://www.ietf.org/rfc/rfc3986.txt (cit. on pp. 9, 10, 13).

Berners-Lee, Tim and Dan Connolly (2008). *Notation3 (N3): A readable RDF syntax*. Tech. rep. W3C. URL: http://www.w3.org/TeamSubmission/n3/ (cit. on p. 52).

Berners-Lee, Tim, James Hendler, and Ora Lassila (2001a). "The Semantic Web." In: *Scientific American* 284.5, pp. 34–43 (cit. on pp. 3, 13).

Berners-Lee, Tim, James Hendler, and Ora Lassila (2001b). "The Semantic Web." In: *Scientific American* 284.5, pp. 34–43. URL: http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21 (cit. on p. 9).

Bischof, Stefan, Stefan Decker, Thomas Krennwallner, Nuno Lopes, and Axel Polleres (2012). "Mapping between RDF and XML with XSPARQL." In: *Journal on Data Semantics* 1.3, pp. 147–185. ISSN: 1861-2032 (cit. on p. 164).

Bizer, Chris, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann (2009). "DBpedia - A Crystallization Point for the Web of Data." In: *Journal of Web Semantics* 7.3, pp. 154–165. DOI: doi:10.1016/j.websem.2009.07.002. URL: http://jens-lehmann.org/files/2009/dbpedia_jws.pdf (cit. on p. 33).

Bizer, Christian (2007). "Quality-Driven Information Filtering in the Context of Web-Based Information Systems." PhD thesis. Freie Universität (cit. on pp. 16, 144).

Bizer, Christian and Richard Cyganiak (2009). "Quality-driven information filtering using the WIQA policy framework." In: *Web Semantics* 7.1, pp. 1 –10 (cit. on pp. 91, 144, 152).

Bizer, Christian, Tom Heath, and Tim Berners-Lee (2009). "Linked Data - The Story So Far." In: *Int. J. Semantic Web Inf. Syst* 5.3, pp. 1–22. URL: http://www.igi-global.com/articles/details.asp?ID=35386 (cit. on p. 3).

Boehm, Barry W. (1981). *Software Engineering Economics*. Prentice Hall PTR. ISBN: 0138221227 (cit. on p. 152).

Bolikowski, Ł. (2009). "Scale-free topology of the interlanguage links in Wikipedia." In: *ArXiv e-prints*. arXiv: 0904.0564 [physics.soc-ph] (cit. on p. 47).

Bratsas, Charalampos, Lazaros Ioannidis, Dimitris Kontokostas, Sören Auer, Christian Bizer, Sebastian Hellmann, and Ioannis Antoniou (2011a). "DBpedia internationalization—a graphical tool for I18n infobox-to-ontology mappings." In: *International Semantic Web Conference Demo (ISWC2011 Demo), October 23-27, 2011, Bonn, Germany* (cit. on pp. xiii, 39).

Bratsas, Charalampos, Spyros Alexiou, Dimitris Kontokostas, Ioannis Parapontis, Ioannis Antoniou, and George Metakides (2011b). "Greek Open Data in the Age of Linked Data: A Demonstration of LOD Internationalization." In: *ACM Web Science Conference, June 14-17, 2011, Koblenz, Germany*. ACM, pp. 1–4 (cit. on p. 39).

Brümmer, Martin, Ciro Baron, Ivan Ermilov, Markus Freudenberg, Dimitris Kontokostas, and Sebastian Hellmann (2014). "DataID: Towards Semantically Rich Metadata for Complex Datasets." In: *Proceedings of the 10th International Conference on Semantic Systems*. SEM '14. Leipzig, Germany: ACM, pp. 84–91. ISBN: 978-1-4503-2927-9. DOI: 10.1145/2660517.2660538. URL: http://svn.aksw.org/papers/2014/Semantics_Dataid/public.pdf (cit. on pp. xii, 32, 66, 79).

Bühmann, Lorenz and Jens Lehmann (2012). "Universal OWL Axiom Enrichment for Large Knowledge Bases." In: *Proceedings of EKAW 2012*. Springer, pp. 57–71. URL: http://jens-lehmann.org/files/2012/ekaw_enrichment.pdf (cit. on p. 95).

Bühmann, Lorenz and Jens Lehmann (2013). "Pattern Based Knowledge Base Enrichment." English. In: *The Semantic Web – ISWC 2013*. Ed. by Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, JosianeXavier Parreira, Lora Aroyo, Natasha Noy, Chris Welty, and Krzysztof Janowicz. Vol. 8218. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 33–48. ISBN: 978-3-642-41334-6. DOI: 10.1007/978-3-642-41335-3_3. URL: http://svn.aksw.org/papers/2013/ISWC_Pattern_Enrichment/public.pdf (cit. on pp. 95, 115).

Cafarella, Michael J., Alon Y. Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang (2008). "WebTables: exploring the power of tables on the web." In: *PVLDB* 1.1, pp. 538–549 (cit. on p. 145).

Cunningham, Hamish (2005). "Information extraction, automatic." In: *Encyclopedia of language and linguistics*, pp. 665–677 (cit. on p. 4).

Cyganiak, Richard, Andreas Harth, and Aidan Hogan (2008). *N-Quads: Extending N-Triples with Context*. Tech. rep. DERI, NUI Galwa. URL: http://sw.deri.org/2008/07/n-quads/ (cit. on p. 53).

Cyganiak, Richard, David Wood, and Markus Lanthaler (2014). *RDF 1.1 Concepts and Abstract Syntax*. W3C Recommendation. https://www.w3.org/TR/rdf11-mt/ (cit. on p. 10).

Cyganiak, Richard, DERI, NUI Galway Dave Reynolds, and Epimorphics Ltd. *The RDF Data Cube Vocabulary*. W3C Recommendation. URL: https://www.w3.org/TR/vocab-data-cube/ (cit. on p. 34).

Das, Souripriya, Seema Sundara, and Richard Cyganiak (2012). *R2RML: RDB to RDF Mapping Language*. Working Group Recommendation. W3C. URL: http://www.w3.org/TR/r2rml/ (cit. on p. 155).

De Vocht, Laurens, Mathias Van Compernolle, Anastasia Dimou, Pieter Colpaert, Ruben Verborgh, Erik Mannens, Peter Mechant, and Rik Van de Walle (2014). "Converging on semantics to ensure local government data reuse." In: *Proceedings of the 5th workshop on Semantics for Smarter Cities (SSC14), 13th International Semantic Web Conference (ISWC)* (cit. on p. 161).

Debattista, Jeremy, Christoph Lange, and Sören Auer (2014). "Representing dataset quality metadata using multi-dimensional views." In: *Proceedings of the 10th International Conference on Semantic Systems*, pp. 92–99. URL: http://doi.acm.org/10.1145/2660517.2660525 (cit. on p. 145).

Debattista, Jeremy, Jurgen Umbrich, Javier D. Fernandez, Anisa Rula, Amrapali Zaveri, Magnus Knuth, and Dimitris Kontokostas, eds. (2016). *Joint Proceedings of the 2nd Workshop on Managing the Evolution and Preservation of the Data Web (MEPDaW 2016) and the 3rd Workshop on Linked Data Quality (LDQ 2016) co-located with 13th European Semantic Web Conference (ESWC 2016)*. (Heraklion, Crete, Greece, Sept. 2, 2014). CEUR Workshop Proceedings 1585. Heraklion. URL: http://ceur-ws.org/Vol-1585/ (cit. on pp. xi, 183).

Demartini, Gianluca, Djellel Difallah, and Philippe Cudré-Mauroux (2012). "ZenCrowd: Leveraging Probabilistic Reasoning and Crowdsourcing Techniques for Large-Scale Entity Linking." In: *21st International Conference on World Wide Web WWW 2012*, pp. 469 –478 (cit. on p. 145).

Demter, Jan, Sören Auer, Michael Martin, and Jens Lehmann (2012). "LODStats—An Extensible Framework for High-performance Dataset Analytics." In: *Proceedings of the EKAW 2012*. Lecture Notes in Computer Science (LNCS) 7603. 29% acceptance rate. Springer. URL: http://svn.aksw.org/papers/2011/RDFStats/public.pdf (cit. on p. 116).

Deutsch, Alin (2009). "FOL Modeling of Integrity Constraints (Dependencies)." In: *Encyclopedia of Database Systems*. Ed. by LING LIU and M.TAMER ÖZSU. Springer US, pp. 1155–1161. ISBN: 978-0-

387-35544-3. DOI: 10.1007/978-0-387-39940-9_980. URL: http://dx.doi.org/10.1007/978-0-387-39940-9_980 (cit. on p. 147).

Dimou, Anastasia, Miel Vander Sande, Pieter Colpaert, Laurens De Vocht, Ruben Verborgh, Erik Mannens, and Rik Van de Walle (2014a). "Extraction & Semantic Annotation of Workshop Proceedings in HTML using RML." In: *Sem. Publ. Challenge of 11th ESWC* (cit. on p. 161).

Dimou, Anastasia, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle (2014b). "RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data." In: *Workshop on Linked Data on the Web* (cit. on p. 155).

Dimou, Anastasia, Dimitris Kontokostas, Markus Freudenberg, Ruben Verborgh, Jens Lehmann, Erik Mannens, Sebastian Hellmann, and Rik Van de Walle (2015). "Assessing and Refining Mappings to RDF to Improve Dataset Quality." In: *Proceedings of the 14th International Semantic Web Conference* (cit. on pp. xii, 151).

Duerst, M. and M. Suignard (2005). *Internationalized Resource Identifiers (IRIs)*. RFC 3987 (Proposed Standard). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc3987.txt (cit. on pp. 9, 13, 50, 53).

Eckle-Kohler, Judith, John P. McCrae, and C. Chiarcos (2014). "lemonUby - a large, interlinked syntactically-rich lexical resources for ontologies." In: *Semantic Web Journal*. URL: http://www.semantic-web-journal.net/content/lemonuby-large-interlinked-syntactically-rich-lexical-resource-ontologies-1 (cit. on p. 125).

Erdmann, M., K.Nakayama, T. Hara, and S. Nishio (2008). "Extraction of Bilingual Terminology from a Multilingual Web-based Encyclopedia." In: *Journal of Information Processing* 16, pp. 68–79 (cit. on p. 46).

Erxleben, Fredo, Michael Günther, Markus Krötzsch, Julian Mendez, and Denny Vrandečić (2014). "Introducing Wikidata to the Linked Data Web." In: *ISWC'14*. LNCS. Springer (cit. on pp. 70, 84).

Etzioni, Oren, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates (2004). "Web-Scale Information Extraction in KnowitAll." In: *Proceedings of the 13th international conference on World Wide Web*. ACM, pp. 100–110 (cit. on p. 87).

Fan, Wenfei (2008). "Dependencies Revisited for Improving Data Quality." In: *Proceedings of the Twenty-seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. PODS '08. Vancouver, Canada: ACM, pp. 159–170. ISBN: 978-1-60558-152-1. DOI: 10.1145/1376916.1376940. URL: http://doi.acm.org/10.1145/1376916.1376940 (cit. on p. 147).

Färber, Michael, Basil Ell, Carsten Menne, Achim Rettinger, and Frederic Bartscherer (2016). "Linked Data Quality of DBpedia, Free-

base, OpenCyc, Wikidata, and YAGO." In: *Semantic Web* Under review (cit. on pp. 69, 146).

Fielding, R., J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee (1999). *Hypertext Transfer Protocol – HTTP/1.1 (RFC 2616)*. Ed. by Internet Engineering Task Force (IETF). RFC 2616 (Proposed Standard). URL: http://www.ietf.org/rfc/rfc2616.txt (cit. on pp. 13, 51).

Flati, Tiziano, Daniele Vannella, Tommaso Pasini, and Roberto Navigli (2014). "Two is Bigger (and Better) Than One: the Wikipedia Bitaxonomy Project." In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics* (cit. on p. 34).

Flemming, Annika (2010). "Quality Characteristics of Linked Data Publishing Datasources." MA thesis. Humboldt-University of Berlin (cit. on pp. 91, 144).

Fossati, Marco, Dimitris Kontokostas, and Jens Lehmann (2015). "Unsupervised Learning of an Extensive and Usable Taxonomy for DBpedia." In: *Proceedings of the 11th International Conference on Semantic Systems*. SEM '15. Vienna, Austria: ACM (cit. on pp. xii, 33–35).

Freudenberg, Markus, Martin Brummer, Jessika Rucknagel, Robert Ulrich, Thomas Eckart, Dimitris Kontokostas, and Sebastian Hellmann (2016). "The Metadata Ecosystem of DataID." In: *Special Track on Metadata & Semantics for Open Repositories at 10th International Conference on Metadata and Semantics Research*. URL: https://svn.aksw.org/papers/2016/MSOR_DataID2/public.pdf (cit. on pp. 32, 33).

Frischmuth, Philipp, Michael Martin, Sebastian Tramp, Thomas Riechert, and Sören Auer (2015). "OntoWiki - An Authoring, Publication and Visualization Interface for the Data Web." In: *Semantic Web Journal* 6.3, pp. 215–240. DOI: 10.3233/SW-140145. URL: http://www.semantic-web-journal.net/system/files/swj490_0.pdf (cit. on p. 129).

Fürber, Christian and Martin Hepp (2010). "Using SPARQL and SPIN for Data Quality Management on the Semantic Web." In: *BIS*. Ed. by Witold Abramowicz and Robert Tolksdorf. Vol. 47. Lecture Notes in Business Information Processing. Springer, pp. 35–46. ISBN: 978-3-642-12813-4. URL: http://dblp.uni-trier.de/db/conf/bis/bis2010.html\#FurberH10 (cit. on p. 147).

Fürber, Christian and Martin Hepp (2010). "Using Semantic Web Resources for Data Quality Management." In: *Knowledge Engineering and Management by the Masses*. Ed. by Philipp Cimiano and H.Sofia Pinto. Vol. 6317. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 211–225. ISBN: 978-3-642-16437-8. DOI: 10.1007/978-3-642-16438-5_15. URL: http://dx.doi.org/10.1007/978-3-642-16438-5_15 (cit. on p. 147).

Gangemi, Aldo, Andrea Giovanni Nuzzolese, Valentina Presutti, Francesco Draicchio, Alberto Musetti, and Paolo Ciancarini (2012). "Automatic Typing of DBpedia Entities." In: *The Semantic Web–ISWC 2012*. Springer, pp. 65–81 (cit. on p. 34).

Gayo, Jose Emilio Labra, Dimitris Kontokostas, and Sören Auer (2013). "Multilingual Linked Data Patterns." In: *Semantic Web Journal* (cit. on pp. xi, 181).

Grant, J. and D. Becket (2004). *RDF Test Cases - N-Triples*. Tech. rep. W3C Recommendation. URL: http://www.w3.org/TR/rdf-testcases/\#ntriples (cit. on p. 52).

Guéret, Christophe, Paul T. Groth, Claus Stadler, and Jens Lehmann (2012). "Assessing Linked Data Mappings Using Network Measures." In: *Proceedings of the 9th Extended Semantic Web Conference*. Vol. 7295. Lecture Notes in Computer Science. Springer, pp. 87–102. URL: http://jens-lehmann.org/files/2012/linked_mapping_qa.pdf (cit. on pp. 91, 144).

Guha, Ramanathan and Dan Brickley (2014). *RDF Schema 1.1*. URL: http://www.w3.org/TR/2014/REC-rdf-schema-20140225/ (cit. on p. 11).

Hayes, Patrick J. and Peter F. Patel-Schneider (2014). *RDF 1.1 Semantics*. W3C Recommendation. https://www.w3.org/TR/rdf11-mt/ (cit. on p. 10).

Hellmann, Sebastian, Jonas Brekle, and Sören Auer (2012). "Leveraging the Crowdsourcing of Lexical Resources for Bootstrapping a Linguistic Data Cloud." In: *JIST*. URL: http://svn.aksw.org/papers/2012/JIST_Wiktionary/public.pdf (cit. on pp. 38, 125, 183).

Hellmann, Sebastian, Claus Stadler, Jens Lehmann, and Sören Auer (2009). "DBpedia Live Extraction." In: *Proc. of 8th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE)*. Vol. 5871. Lecture Notes in Computer Science, pp. 1209–1223. DOI: doi:10.1007/978-3-642-05151-7_33. URL: http://svn.aksw.org/papers/2009/ODBASE_LiveExtraction/dbpedia_live_extraction_public.pdf (cit. on p. 31).

Hellmann, Sebastian, Jens Lehmann, Sören Auer, and Martin Brümmer (2013a). "Integrating NLP using Linked Data." In: *12th International Semantic Web Conference, 21-25 October 2013, Sydney, Australia*. URL: http://svn.aksw.org/papers/2013/ISWC_NIF/public.pdf (cit. on pp. 111, 125).

Hellmann, Sebastian, Agata Filipowska, Caroline Barriere, Pablo Mendes, and Dimitris Kontokostas, eds. (2013b). *Proceedings of the NLP and DBpedia Workshop in conjunction with the 12th International Semantic Web Conference (ISWC 2013) Sydney, Australia, October, 2013*. Vol. 1064. CEUR Workshop Proceedings. CEUR-WS.org. URL: http://ceur-ws.org/Vol-1064 (cit. on pp. xi, 185).

Hellmann, Sebastian, Volha Bryl, Lorenz B̈hmann, Milan Dojchinovski, Dimitris Kontokostas, Jens Lehmann, Uroš Milošević, Petar Petrovski, Vojtěch Svátek, Mladen Stanojević, and Ondřej Zamazal (2014). "Knowledge Base Creation, Enrichment and Repair." English. In: *Linked Open Data – Creating Knowledge Out of Interlinked Data*. Ed. by Sören Auer, Volha Bryl, and Sebastian Tramp. Lecture Notes in Computer Science. Springer International Publishing, pp. 45–69. ISBN: 978-3-319-09845-6. DOI: 10.1007/978-3-319-09846-3_3. URL: http://dx.doi.org/10.1007/978-3-319-09846-3_3 (cit. on p. xii).

Hernandez, Daniel, Aidan Hogan, and Markus Krötzsch (2015). "Reifying RDF: What Works Well With Wikidata?" In: *SSWS@ISWC*. Ed. by Thorsten Liebig and Achille Fokoue. Vol. 1457. CEUR Workshop Proceedings. CEUR-WS.org, pp. 32–47. URL: http://dblp.uni-trier.de/db/conf/semweb/ssws2015.html#HernandezHK15 (cit. on p. 70).

Hert, Matthias, Gerald Reif, and Harald C. Gall (2011). "A comparison of RDB-to-RDF mapping languages." In: I-Semantics '11. ACM, pp. 25–32 (cit. on p. 165).

Heyvaert, Pieter, Anastasia Dimou, Ruben Verborgh, Erik Mannens, and Rik Van de Walle (2015). "Extraction and Semantic Annotation of Workshop Proceedings in HTML using RML." In: *Semantic Publishing Challenge of the 12th ESWC* (cit. on p. 161).

Hitzler, P., M. Krötzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph (2009). *OWL 2 Web Ontology Language Primer. W3C Recommendation 27 October 2009*. URL: http://www.w3.org/TR/owl2-primer/ (cit. on p. 11).

Hoffart, Johannes, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum (2013a). "YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia." In: *Artif. Intell* 194, pp. 28–61. URL: http://dx.doi.org/10.1016/j.artint.2012.06.001 (cit. on p. 85).

Hoffart, Johannes, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum (2013b). "YAGO2: a Spatially and Temporally Enhanced Knowledge Base from Wikipedia." In: *AI* 194, pp. 28–61 (cit. on p. 34).

Hogan, Aidan, Andreas Harth, Alexandre Passant, Stefan Decker, and Axel Polleres (2010). "Weaving the Pedantic Web." In: *LDOW* (cit. on p. 145).

Hogan, Aidan, Jürgen Umbrich, Andreas Harth, Richard Cyganiak, Axel Polleres, and Stefan Decker (2012). "An empirical survey of Linked Data conformance." In: *Journal of Web Semantics* 14, pp. 14–44 (cit. on p. 145).

Holtman, K. and A. Mutz (1998). *Transparent Content Negotiation in HTTP*. RFC 2295 (Experimental). Internet Engineering Task Force.

URL: http://www.ietf.org/rfc/rfc2295.txt (cit. on pp. 13, 39, 50).

Hyland, Bernard, Ghislain Atemezing, and Boris Villazon-Terrazas (2004). *Best Practices for Publishing Linked Data*. Working Group Note. W3C. URL: http://www.w3.org/TR/ld-bp/ (cit. on p. 151).

ISO/IEC (2015). *ISO/IEC 2382. Information technology - Vocabulary*. ISO/IEC (cit. on p. 15).

Ismayilov, Ali, Dimitris Kontokostas, Sören Auer, Jens Lehmann, and Sebastian Hellmann (2016). "Wikidata through the Eyes of DBpedia (to appear)." In: *Semantic Web Journal*. http://www.semantic-web-journal.net/content/wikidata-through-eyes-dbpedia (cit. on pp. xi, 33, 67).

Ji, Qiu, Peter Haase, Guilin Qi, Pascal Hitzler, and Steffen Stadtmüller (2009). "RaDON - Repair and Diagnosis in Ontology Networks." In: *ESWC*. Ed. by Lora Aroyo, Paolo Traverso, Fabio Ciravegna, Philipp Cimiano, Tom Heath, Eero Hyvönen, Riichiro Mizoguchi, Eyal Oren, Marta Sabou, and Elena Paslaru Bontas Simperl. Vol. 5554. Lecture Notes in Computer Science. Springer, pp. 863–867. ISBN: 978-3-642-02120-6 (cit. on p. 91).

Juran, Joseph (1974). *The Quality Control Handbook*. McGraw-Hill, New York (cit. on pp. 4, 91, 130, 152).

Knublauch, Holger, James A. Hendler, and Kingsley Idehen (2011). *SPIN - Overview and Motivation*. W3C Member Submission, W3C. URL: http://www.w3.org/Submission/2011/SUBM-spin-overview-20110222/ (cit. on pp. 98, 143).

Knublauch, Holger and Dimitris Kontokostas (2016). *Shapes Constraint Language (SHACL)*. Public Working Draft. W3C. URL: http://www.w3.org/TR/shacl (cit. on pp. xi, 144).

Knuth, Magnus, Johannes Hercher, and Harald Sack (2012). "Collaboratively Patching Linked Data." In: *CoRR* (cit. on pp. 132, 141).

Knuth, Magnus, Dimitris Kontokostas, and Harald Sack, eds. (2014). *Proceedings of the 1st Workshop on Linked Data Quality (LDQ)*. (Leipzig, Germany, Sept. 2, 2014). CEUR Workshop Proceedings 1215. Aachen. URL: http://ceur-ws.org/Vol-1215/ (cit. on pp. xi, 183).

Kobilarov, Georgi, Christian Bizer, Sören Auer, and Jens Lehmann (2009). "DBpedia - A Linked Data Hub and Data Source for Web Applications and Enterprises." In: *Proceedings of Developers Track of 18th International World Wide Web Conference (WWW 2009), April 20th-24th, Madrid, Spain*. URL: http://www2009.eprints.org/228/ (cit. on p. 3).

Kontokostas, Dimitris and Sebastian Hellmann (2014). "The DBpedia Data Stack—Towards a sustainable DBpedia Project to provide a public data infrastructure for Europe." In: *EDF2014 Poster Session*. URL: http://svn.aksw.org/papers/2014/EDF_DBpediaDataStack/public.pdf (cit. on pp. xii, 19).

Kontokostas, Dimitris, Charalampos Bratsas, Sören Auer, Sebastian Hellmann, Ioannis Antoniou, and George Metakides (2011). "Towards Linked Data Internationalization - Realizing the Greek DBpedia." In: *Proceedings of the ACM WebSci'11*. URL: http://journal.webscience.org/475/ (cit. on p. xiii).

Kontokostas, Dimitris, Charalampos Bratsas, Sören Auer, Sebastian Hellmann, Ioannis Antoniou, and George Metakides (2012). "Internationalization of Linked Data: The case of the Greek DBpedia edition." In: *Web Semantics: Science, Services and Agents on the World Wide Web* 15.0, pp. 51 –61. ISSN: 1570-8268. DOI: 10.1016/j.websem.2012.01.001. URL: http://www.sciencedirect.com/science/article/pii/S1570826812000030 (cit. on pp. xii, 26, 27, 31, 33, 39, 61, 116, 146).

Kontokostas, Dimitris, Amrapali Zaveri, Sören Auer, and Jens Lehmann (2013). "TripleCheckMate: A Tool for Crowdsourcing the Quality Assessment of Linked Data." In: *Proceedings of the 4th Conference on Knowledge Engineering and Semantic Web*. URL: http://jens-lehmann.org/files/2013/kesw_triplecheckmate.pdf (cit. on pp. xiii, 129).

Kontokostas, Dimitris, Patrick Westphal, Sören Auer, Sebastian Hellmann, Jens Lehmann, and Roland Cornelissen (2014a). "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. Seoul, Korea: International World Wide Web Conferences Steering Committee, pp. 115–118. ISBN: 978-1-4503-2745-9. DOI: 10.1145/2567948.2577017. URL: http://jens-lehmann.org/files/2014/www_demo_databugger.pdf (cit. on pp. xiii, 111, 173).

Kontokostas, Dimitris, Martin Brümmer, Sebastian Hellmann, Jens Lehmann, and Lazaros Ioannidis (2014b). "NLP data cleansing based on Linguistic Ontology constraints." In: *Proc. of the Extended Semantic Web Conference 2014*. URL: http://jens-lehmann.org/files/2014/eswc_rdfunit_nlp.pdf (cit. on pp. xii, 91, 111, 121, 143, 155, 158, 174).

Kontokostas, Dimitris, Patrick Westphal, Sören Auer, Sebastian Hellmann, Jens Lehmann, Roland Cornelissen, and Amrapali Zaveri (2014c). "Test-driven Evaluation of Linked Data Quality." In: *Proceedings of the 23rd International Conference on World Wide Web*. WWW '14. Seoul, Korea: International World Wide Web Conferences Steering Committee, pp. 747–758. ISBN: 978-1-4503-2744-2. DOI: 10.1145/2566486.2568002. URL: http://svn.aksw.org/papers/2014/WWW_Databugger/public.pdf (cit. on pp. xii, 91, 111, 115, 143, 151).

Kontokostas, Dimitris, Christian Mader, Christian Dirschl, Katja Eck, Michael Leuthold, Jens Lehmann, and Sebastian Hellmann (2016). "Semantically Enhanced Quality Assurance in the JURION Busi-

ness Use Case." In: *13th International Conference, ESWC 2016, Heraklion, Crete, Greece, May 2016*, pp. 661–676. ISBN: 978-3-319-34129-3. DOI: 10.1007/978-3-319-34129-3_40. URL: http://svn.aksw.org/papers/2016/ESWC_Jurion/public.pdf (cit. on pp. xii, 167).

Kreis, Paul (2011). "Design of a Quality Assessment Framework for the DBpedia Knowledge Base." MA thesis. Freie Universität Berlin (cit. on p. 145).

Krejcie and Morgan (1970). "Determining Sample Size for Research Activities." In: *Educational and Psycholoigcal Measurement* 30, pp. 607–610 (cit. on p. 139).

Krötzsch, Markus, Denny Vrandečić, Max Völkel, Heiko Haller, and Rudi Studer (2007). "Semantic Wikipedia." In: *Journal of Web Semantics* 5, pp. 251–261 (cit. on p. 129).

Langegger, Andreas and Wolfram Wöß (2009). "XLWrap – Querying and Integrating Arbitrary Spreadsheets with SPARQL." In: *Proceedings of 8th ISWC*. Springer, pp. 359–374 (cit. on p. 164).

Lausen, Georg, Michael Meier, and Michael Schmidt (2008). "SPARQLing Constraints for RDF." In: *Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology*. EDBT '08. Nantes, France: ACM, pp. 499–509. ISBN: 978-1-59593-926-5. DOI: 10.1145/1353343.1353404. URL: http://doi.acm.org/10.1145/1353343.1353404 (cit. on p. 147).

Lehmann, Jens, Chris Bizer, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann (2009). "DBpedia - A Crystallization Point for the Web of Data." In: *Journal of Web Semantics* 7.3, pp. 154–165. DOI: doi:10.1016/j.websem.2009.07.002. URL: http://jens-lehmann.org/files/2009/dbpedia_jws.pdf (cit. on pp. 3, 47, 138, 146).

Lehmann, Jens, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer (2015). "DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia." In: *Semantic Web Journal* 6.2, pp. 167–195. URL: http://jens-lehmann.org/files/2014/swj_dbpedia.pdf (cit. on pp. xi, 19, 59, 67, 83, 102, 115, 118, 160, 183).

Lenat, Douglas B (1995). "CYC: A Large-Scale Investment in Knowledge Infrastructure." In: *Communications of the ACM* 38.11, pp. 33–38 (cit. on p. 87).

Lukovnikov, Denis, Claus Stadler, Dimitris Kontokostas, Sebastian Hellmann, and Jens Lehmann (2014). "DBpedia Viewer - An Integrative Interface for DBpedia leveraging the DBpedia Service Eco System." In: *Proc. of the Linked Data on the Web 2014 Workshop*. URL: http://jens-lehmann.org/files/2014/ldow_dbpedia_viewer.pdf (cit. on pp. xii, 35, 65).

Maali, Fadi, DERI, and NUI Galway. *Data Catalog Vocabulary (DCAT)*. W3C Recommendation. URL: https://www.w3.org/TR/vocab-dcat/ (cit. on p. 32).

Manola, Frank and Eric Miller, eds. (2004). *RDF Primer*. W3C Recommendation. World Wide Web Consortium. URL: http://www.w3.org/TR/rdf-primer/ (cit. on p. 53).

McCrae, John, Guadalupe Aguado-de Cea, Paul Buitelaar, Philipp Cimiano, Thierry Declerck, Asuncion Gomez-Perez, Jorge Gracia, Laura Hollink, Elena Montiel-Ponsoda, Dennis Spohr, and Tobias Wunner (2012). "Interchanging lexical resources on the Semantic Web." English. In: *LRE* 46.4, pp. 701–719. ISSN: 1574-020X. DOI: 10.1007/s10579-012-9182-3. URL: http://dx.doi.org/10.1007/s10579-012-9182-3 (cit. on p. 111).

McGuinness, Deborah, Timothy Lebo, and Satya Sahoo. *The PROV Ontology*. W3C Recommendation. URL: http://www.w3.org/TR/prov-o/ (cit. on p. 33).

Melo, Gerard de and Gerhard Weikum (2010). "MENTA: Inducing Multilingual Taxonomies from Wikipedia." In: *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*. ACM, pp. 1099–1108 (cit. on p. 34).

Mendes P.N. Mühleisen H., Bizer C. (2012). "Sieve: Linked Data Quality Assessment and Fusion." In: *LWDM* (cit. on pp. 91, 144).

Meyer, Scott M., Jutta Degener, John Giannandrea, and Barak Michener (2010). "Optimizing Schema-Last Tuple-Store Queries in GraphD." In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010*. Ed. by Ahmed K. Elmagarmid and Divyakant Agrawal. ACM, pp. 1047–1056. ISBN: 978-1-4503-0032-2. URL: http://doi.acm.org/10.1145/1807167.1807283 (cit. on p. 85).

Moran, Steven and Martin Brümmer (2013). "Lemon-aid: using Lemon to aid quantitative historical linguistic analysis." In: *LDL* (cit. on p. 125).

Nakayama, Kotaro, Minghua Pei, Maike Erdmann, Masahiro Ito, Masumi Shirakawa, Takahiro Hara, and Shojiro Nishio (2008). "Wikipedia Mining: Wikipedia as a Corpus for Knowledge Extraction." In: *Annual Wikipedia Conference (Wikimania)* (cit. on p. 86).

Nastase, Vivi and Michael Strube (2013). "Transforming Wikipedia into a Large Scale Multilingual Concept Network." In: *Artificial Intelligence* 194, pp. 62–85 (cit. on p. 34).

Nastase, Vivi, Michael Strube, Benjamin Boerschinger, Cäcilia Zirn, and Anas Elghafari (2010). "WikiNet: A Very Large Scale Multi-Lingual Concept Network." In: *Proceedings of the 5th Language Resources and Evaluation Conference* (cit. on p. 34).

Niu, Xing, Xinruo Sun, Haofen Wang, Shu Rong, Guilin Qi, and Yong Yu (2011). "Zhishi.me: Weaving Chinese Linking Open Data." In: *10th International Conference on The semantic web - Volume Part II*.

Bonn, Germany: Springer-Verlag, pp. 205–220. ISBN: 978-3-642-25092-7. URL: http://dl.acm.org/citation.cfm?id=2063076.2063091 (cit. on p. 87).

O'Connor, Martin J., Christian Halaschek-Wiener, and Mark A. Musen (2010). "Mapping Master: a flexible approach for mapping spreadsheets to OWL." In: ISWC'10 (cit. on p. 164).

Olson, David L. and Dursun Delen (2008). *Advanced Data Mining Techniques*. Springer, pp. I–XII, 1–180. ISBN: 978-3-540-76916-3 (cit. on p. 4).

Paulheim, Heiko (2016). "Knowledge graph refinement: A survey of approaches and evaluation methods." In: *Semantic Web* Preprint, pp. 1–20 (cit. on pp. 69, 147).

Paulheim, Heiko and Christian Bizer (2013). "Type Inference on Noisy RDF Data." In: *The Semantic Web–ISWC 2013*. Springer, pp. 510–525 (cit. on p. 34).

Pohl, Aleksander (2012). "Classifying the Wikipedia Articles into the OpenCyc Taxonomy." In: *Proceedings of the Web of Linked Entities Workshop in conjuction with the 11th International Semantic Web Conference* (cit. on p. 34).

Ponzetto, Simone Paolo and Michael Strube (2007). "Deriving a Large Scale Taxonomy from Wikipedia." In: *Proceeding of AAAI*. Vol. 7, pp. 1440–1445 (cit. on p. 34).

Ponzetto, Simone Paolo and Michael Strube (2008). "WikiTaxonomy: A Large Scale Knowledge Resource." In: *ECAI 2008 - 18th European Conference on Artificial Intelligence, Patras, Greece, July 21-25, 2008, Proceedings*. Ed. by Malik Ghallab, Constantine D. Spyropoulos, Nikos Fakotakis, and Nikolaos M. Avouris. Vol. 178. Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 751–752. ISBN: 978-1-58603-891-5. URL: http://www.booksonline.iospress.nl/Content/View.aspx?piid=9905 (cit. on p. 87).

Ponzetto, Simone Paolo and Michael Strube (2011). "Taxonomy Induction Based on a Collaboratively Built Knowledge Repository." In: *Artificial Intelligence* 175.9-10, pp. 1737–1756 (cit. on p. 34).

SPARQL 1.1 Query Language (2013). Tech. rep. W3C. URL: http://www.w3.org/TR/sparql11-query (cit. on pp. 12, 13, 49, 51).

Prud'hommeaux, Eric, Jose Emilio Labra Gayo, and Harold Solbrig (2014). "Shape Expressions: An RDF Validation and Transformation Language." In: *Proceedings of the 10th International Conference on Semantic Systems*. ACM, pp. 32–40. URL: http://doi.acm.org/10.1145/2660517.2660523 (cit. on p. 144).

Röder, Michael, Ricardo Usbeck, Sebastian Hellmann, Daniel Gerber, and Andreas Both (2014). "N3 - A Collection of Datasets for Named Entity Recognition and Disambiguation in the NLP Interchange Format." In: *The 9th edition of the Language Resources and Evaluation Conference, 26-31 May, Reykjavik, Iceland*. URL: http:

//svn.aksw.org/papers/2014/LREC_N3NIFNERNED/public.pdf (cit. on pp. 124, 125).

Roth, C, D Taraborelli, and N Gilbert (2008). "Measuring Wiki viability: An empirical assessment of the social dynamics of a large sample of Wikis." In: *WikiSym 2008 - The 4th International Symposium on Wikis, Proceedings*. URL: http://epubs.surrey.ac.uk/1565/ (cit. on p. 4).

Rula, Anisa, Amrapali Zaveri, Magnus Knuth, and Dimitris Kontokostas, eds. (2015). *Proceedings of the 2nd Workshop on Linked Data Quality (LDQ)*. (Portorož, Slovenia, Sept. 2, 2014). CEUR Workshop Proceedings 1376. Portorož. URL: http://ceur-ws.org/Vol-1376/ (cit. on pp. xi, 183).

Sarasua, Cristina, Elena Simperl, and NatalyaF. Noy (2012). "CrowdMap: Crowdsourcing Ontology Alignment with Microtasks." In: *The Semantic Web – ISWC 2012*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 525–541 (cit. on p. 145).

Sauermann, Leo and Richard Cyganiak (2008). *Cool URIs for the Semantic Web*. W3C Interest Group Note. URL: http://www.w3.org/TR/cooluris/ (cit. on pp. 13, 14, 50, 52).

Schmachtenberg, Max, Christian Bizer, and Heiko Paulheim (2014). "Adoption of the Linked Data Best Practices in Different Topical Domains." In: vol. 8796. LNCS. Springer (cit. on p. 151).

Siorpaes, Katharina and Martin Hepp (2008). "Games with a Purpose for the Semantic Web." In: *IEEE Intelligent Systems* 23.3, pp. 50–60. ISSN: 1541-1672. DOI: http://doi.ieeecomputersociety.org/10.1109/MIS.2008.45 (cit. on p. 129).

Sirin, Evren and Jiao Tao (2009). "Towards integrity constraints in OWL." In: *Proceedings of the Workshop on OWL: Experiences and Directions, OWLED* (cit. on pp. 93, 147).

Stadler, Claus, Jens Lehmann, Konrad Höffner, and Sören Auer (2012). "LinkedGeoData: A Core for a Web of Spatial Open Data." In: *Semantic Web Journal* 3.4, pp. 333–354. URL: http://jens-lehmann.org/files/2012/linkedgeodata2.pdf (cit. on pp. 38, 102, 116, 183).

Steinmetz, Nadine, Magnus Knuth, and Harald Sack (2013). "Statistical Analyses of Named Entity Disambiguation Benchmarks." In: *Proceedings of 1st International Workshop on NLP and DBpedia, October 21-25, Sydney, Australia*. Vol. 1064. NLP & DBpedia 2013. Sydney, Australia: CEUR Workshop Proceedings (cit. on p. 125).

Suchanek, Fabian M, Gjergji Kasneci, and Gerhard Weikum (2007). "Yago: a core of semantic knowledge." In: *Proceedings of the 16th international conference on World Wide Web (WWW)*. ACM, pp. 697–706 (cit. on p. 85).

Suominen, Osma and Eero Hyvönen (2012). "Improving the quality of SKOS vocabularies with skosify." In: *Proceedings of the 18th international conference on Knowledge Engineering and Knowledge Manage-*

ment. EKAW'12. Galway City, Ireland: Springer-Verlag, pp. 383–397. ISBN: 978-3-642-33875-5. DOI: 10.1007/978-3-642-33876-2_34. URL: http://dx.doi.org/10.1007/978-3-642-33876-2_34 (cit. on pp. 115, 118).

Tacchini, Eugenio, Andreas Schultz, and Christian Bizer (2009). "Experiments with Wikipedia Cross-Language Data Fusion." In: *Proceedings of the 5th Workshop on Scripting and Development for the Semantic Web, co-located with ESWC* (cit. on p. 25).

Tanon, Thomas Pellissier, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher (2016). "From Freebase to Wikidata: The Great Migration." In: *World Wide Web Conference* (cit. on p. 71).

Troncy, Raphaël, Erik Mannens, Silvia Pfeiffer, and Davy Van Deursen (2012). *Media Fragments URI*. W3C Recommendation. URL: http://www.w3.org/TR/media-frags/ (cit. on p. 62).

Vaidya, Gaurav, Dimitris Kontokostas, Magnus Knuth, Jens Lehmann, and Sebastian Hellmann (2015). "DBpedia Commons: Structured Multimedia Metadata from the Wikimedia Commons." In: *Proceedings of the 14th International Semantic Web Conference*. URL: http://svn.aksw.org/papers/2015/ISWCData_DBpediaCommons/public.pdf (cit. on pp. xii, 33, 59, 67).

Vrandečić, Denny (2010). "Ontology Evaluation." PhD thesis. Karlsruhe: KIT, Fakultät für Wirtschaftswissenschaften (cit. on p. 45).

Vrandečić, Denny and Markus Krötzsch (2014). "Wikidata: A Free Collaborative Knowledgebase." In: *Commun. ACM* 57.10, pp. 78–85. ISSN: 0001-0782. DOI: 10.1145/2629489. URL: http://doi.acm.org/10.1145/2629489 (cit. on pp. xxv, 67, 84).

Wang, Jiannan, Tim Kraska, Michael J. Franklin, and Jianhua Feng (2012). "CrowdER: crowdsourcing entity resolution." In: *Proc. VLDB Endow.* 5, pp. 1483–1494 (cit. on pp. 87, 145).

Wu, Fei and Daniel S. Weld (2007). "Autonomously Semantifying Wikipedia." In: *Proceedings of the 16th Conference on Information and Knowledge Management*. ACM, pp. 41–50 (cit. on pp. 86, 87).

Wu, Fei and Daniel S. Weld (2008). "Automatically Refining the Wikipedia Infobox Ontology." In: *Proceedings of the 17th International Conference on World Wide Web*. WWW '08. Beijing, China: ACM, pp. 635–644. ISBN: 978-1-60558-085-2. DOI: 10.1145/1367497.1367583. URL: http://doi.acm.org/10.1145/1367497.1367583 (cit. on p. 87).

Zaveri, Amrapali, Dimitris Kontokostas, Mohamed Ahmed Sherif, Lorenz Bühmann, Mohamed Morsey, Sören Auer, and Jens Lehmann (2013). "User-driven Quality Evaluation of DBpedia." In: *Proceedings of 9th International Conference on Semantic Systems, I-SEMANTICS '13, Graz, Austria, September 4-6, 2013*. ACM, pp. 97–104. URL: http://svn.aksw.org/papers/2013/ISemantics_DBpediaDQ/public.pdf (cit. on pp. xii, 129, 133, 143, 146).

Zaveri, Amrapali, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann, and Sören Auer (2015). "Quality Assessment for

Linked Data: A Survey." In: *Semantic Web Journal*. URL: http://www.semantic-web-journal.net/content/quality-assessment-linked-data-survey (cit. on pp. 16, 132, 138, 151, 152).

Zesch, Torsten, Christof Müller, and Iryna Gurevych (2008). "Extracting Lexical Semantic Knowledge from Wikipedia and Wiktionary." In: *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC)*. Vol. 8, pp. 1646–1652 (cit. on p. 86).

Zhu, Hong, Patrick A. V. Hall, and John H. R. May (1997). "Software Unit Test Coverage and Adequacy." In: *ACM Comput. Surv.* 29.4, pp. 366–427. URL: http://dblp.uni-trier.de/db/journals/csur/csur29.html\#ZhuHM97 (cit. on pp. 91, 96).

Curriculum Vitae

Dimitris Kontokostas

Education and Employment

07-2015–present **Scientific Assistant**, *AKSW/KILT, University of Leipzig, Germany.*

02-2015–present **WP leader of the ALIGNED H2020 project**, *AKSW/KILT, University of Leipzig, Germany.*

11-2014–present **Member of the RDF Data Shapes WG**, *W3C.*

04-2014–present **Head of technical developments**, *DBpedia Association.*

09/2012–06/2015 **Researcher at the AKSW/KILT research group**, *AKSW/KILT, University of Leipzig, Germany.*

04/2012–08/2012 **R&D Software Engineer**, *Semantic Web Unit / Aristotle University of Thessaloniki.*

09/2009–05/2012 **MSc in Web Science**, *Aristotle University of Thessaloniki.*

09/2006–08/2011 **ICT Teacher**, *Ministry of Education, Greece.*

02/2005–06/2006 **R&D Software Engineer**, *AIIA Lab / Aristotle University of Thessaloniki.*

05/2005–05/2006 **Obligatory military service**, *Ministry of Defence, Greece.*

06/2001–07/2003 **R&D Software Engineer**, *MUSIC Lab / Technical University of Crete.*

09/1999–07/2004 **Diploma in Electronic & Computer Engineering**, *Technical University of Crete.*

kontokostas@informatik.uni-leipzig.de
http://aksw.org/DimitrisKontokostas
June 06, 1981 in Veria, Greece

Organisation and Chairing of Scientific Events

Jun 2016 **LDQ 2016**, *http://ldq.semanticmultimedia.org/sites/archive/2016.ldq.semanticmultimedia.org/*, Heraklion, Greece.

Joint Proceedings of the 2nd Workshop on Managing the Evolution and Preservation of the Data Web (MEPDaW 2016) and the 3rd Workshop on Linked Data Quality (LDQ 2016) co-located with 13th European Semantic Web Conference (ESWC 2016). Heraklion, Crete, Greece. Debattista, Jeremy, Jurgen Umbrich, Javier D. Fernandez, Anisa Rula, Amrapali Zaveri, Magnus Knuth, and Dimitris Kontokostas (Eds.). Volume 1585 of CEUR Workshop Proceedings, `http://ceur-ws.org/Vol-1585` – 3 accepted papers by 9 authors, 20 PC members

Jun 2015 **LDQ 2015**, *http://ldq.semanticmultimedia.org/sites/archive/2015.ldq.semanticmultimedia.org/*, Portoroz, Slovenia.

Proceedings of the 2nd Workshop on Linked Data Quality co-located with 12th Extended Semantic Web Conference (ESWC 2015) Portoroz, Slovenia, June 1, 2015. Anisa Rula, Amrapali Zaveri, Magnus Knuth, Dimitris Kontokostas (Eds.). Volume 1376 of CEUR Workshop Proceedings, `http://ceur-ws.org/Vol-1376` – 7 accepted papers by 21 authors, 25 PC members

Sep 2014 **LDQ 2014**, *http://ldq.semanticmultimedia.org/sites/archive/2014.ldq.semanticmultimedia.org/*, Leipzig, Germany.

Proceedings of the 1st Workshop on Linked Data Quality co-located with 10th International Conference on Semantic Systems (SEMANTiCS 2014). Leipzig, Germany, September 2nd, 2014.. Magnus Knuth, Dimitris Kontokostas, Harald Sack (Eds.). Volume 1215 of CEUR Workshop Proceedings, `http://ceur-ws.org/Vol-1215` – 6 accepted papers by 18 authors, 16 PC members

Oct 2013 **NLP & DBpedia 2013**, *http://nlp-dbpedia2013.blogs.aksw.org/*, Sydney, Australia.

Proceedings of the NLP and DBpedia Workshop in conjunction with the 12th International Semantic Web Conference (ISWC 2013) Sydney, Australia, October, 2013. Sebastian Hellmann, Agata Filipowska, Caroline Barriere, Pablo Mendes, and Dimitris Kontokostas (Eds.). Volume 1064 of CEUR Workshop Proceedings, `http://ceur-ws.org/Vol-1064` – 11 accepted papers by 31 authors, 29 PC members

Community Service (Selection)

kontokostas@informatik.uni-leipzig.de
http://aksw.org/DimitrisKontokostas
June 06, 1981 in Veria, Greece

2012-2017 **PC Membership and Reviewing**.
International Semantic Web Conference (ISWC), Workd Wide Web Conference (WWW), SEMANTiCS, I-SEMANTICS, I-Challenge, Linked Data on the Web (LDOW), Semantic Web Journal, Journal of Web Semantics, Journal of Data and Information Quality, Journal of the Americal Society for Information Science and Technology,

Presentation and Talks

Nov 2016 15 min. **DBpedia state of affairs**, *http://wiki.dbpedia.org/meetings/California2016*, 8th DBpedia community meeting, Sunnyvale, California, CA.

Sep 2016 45 min. **Introduction to SHACL**, *https://2016.semantics.cc/satellite-events/data-quality-tutorial*, Data Quality Tutorial, SEMANTiCS conference, Leipzig, Germany.

May 2016 45 min. **Graph Databases and Data Integration - the Case of RDF**, *https://www.meetup.com/Thessaloniki-Java-Meetup-Group/events/229442859/*, Data-Driven Night, Thessaloniki Java Meetup, Thessaloniki, Greece.

Jun 2016 30 min. **Semantically enhanced quality assurance in the JURION business case**, –, ESWC 2016 presentation, Heraklion, Greece.

Jun 2015 30 min. **DBpedia past, present & future**, *http://wiki.dbpedia.org/meetings/Poznan2015*, 4th DBpedia community meeting, Poznan, Poland.

Jun 2014 30 min. **NLP Data Cleansing Based on Linguistic Ontological Constraints**, –, ESWC 2014 presentation, Heraklion, Greece.

Apr 2014 30 min. **Test-Driven Data Quality Assessment**, –, WWW 2014 presentation, Seul, Korea.

Apr 2014 30 min. **DBpedia Viewer - An integrative interface for DBpedia leveraging the DBpedia service eco-system**, –, LDOW 2014 presentation, Seul, Korea.

Publications

[1] Maribel Acosta, Amrapali Zaveri, Elena Simperl, Dimitris Kontokostas, Sören Auer, and Jens Lehmann. Crowdsourcing linked data quality assessment. In *12th International Semantic Web Conference, 21-25 October 2013, Sydney, Australia*, pages 260–

kontokostas@informatik.uni-leipzig.de
http://aksw.org/DimitrisKontokostas
June 06, 1981 in Veria, Greece

276, 2013.

[2] Sören Auer and Dimitris Kontokostas. Towards web intelligence through the crowdsourcing of semantics. In *Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web Companion*, WWW Companion '14, pages 991–992, 2014.

[3] Charalampos Bratsas, Lazaros Ioannidis, Dimitris Kontokostas, Sören Auer, Christian Bizer, Sebastian Hellmann, and Ioannis Antoniou. DBpedia internationalization—a graphical tool for I18n infobox-to-ontology mappings. In *International Semantic Web Conference Demo (ISWC2011 Demo), October 23-27, 2011, Bonn, Germany*, 2011.

[4] Martin Brümmer, Ciro Baron, Ivan Ermilov, Markus Freudenberg, Dimitris Kontokostas, and Sebastian Hellmann. DataID: Towards semantically rich metadata for complex datasets. In *Proceedings of the 10th International Conference on Semantic Systems*, SEM '14, pages 84–91. ACM, 2014.

[5] Jeremy Debattista, Jurgen Umbrich, Javier D. Fernandez, Anisa Rula, Amrapali Zaveri, Magnus Knuth, and Dimitris Kontokostas, editors. *Joint Proceedings of the 2nd Workshop on Managing the Evolution and Preservation of the Data Web (MEPDaW 2016) and the 3rd Workshop on Linked Data Quality (LDQ 2016) co-located with 13th European Semantic Web Conference (ESWC 2016)*, number 1585 in CEUR Workshop Proceedings, Heraklion, 2016.

[6] Anastasia Dimou, Dimitris Kontokostas, Markus Freudenberg, Ruben Verborgh, Jens Lehmann, Erik Mannens, Sebastian Hellmann, and Rik Van de Walle. Assessing and Refining Mappings to RDF to Improve Dataset Quality. In *Proceedings of the 14th International Semantic Web Conference*, October 2015.

[7] Milan Dojchinovski, Dimitris Kontokostas, Robert Rößling, Magnus Knuth, and Sebastian Hellmann. Dbpedia links: The hub of links for the web of data. In *Proceedings of the SEMANTiCS 2016 Conference (SEMANTiCS 2016)*, September 2016.

kontokostas@informatik.uni-leipzig.de
http://aksw.org/DimitrisKontokostas
June 06, 1981 in Veria, Greece

[8] Marco Fossati, Dimitris Kontokostas, and Jens Lehmann. Unsupervised learning of an extensive and usable taxonomy for dbpedia. In *Proceedings of the 11th International Conference on Semantic Systems*, SEM '15. ACM, September 2015.

[9] Markus Freudenberg, Martin Brummer, Jessika Rucknagel, Robert Ulrich, Thomas Eckart, Dimitris Kontokostas, and Sebastian Hellmann. The metadata ecosystem of dataid. In *Special Track on Metadata & Semantics for Open Repositories at 10th International Conference on Metadata and Semantics Research*, 2016.

[10] Jose Emilio Labra Gayo, Dimitris Kontokostas, and Sören Auer. Multilingual linked data patterns. *Semantic Web Journal*, 2013.

[11] Sebastian Hellmann, Volha Bryl, Lorenz Bühmann, Milan Dojchinovski, Dimitris Kontokostas, Jens Lehmann, Uroš Milošević, Petar Petrovski, Vojtěch Svátek, Mladen Stanojević, and Ondrej Zamazal. Knowledge base creation, enrichment and repair. In *Linked Open Data–Creating Knowledge Out of Interlinked Data*, pages 45–69. Springer, 2014.

[12] Sebastian Hellmann, Agata Filipowska, Caroline Barriere, Pablo Mendes, and Dimitris Kontokostas, editors. *Proceedings of the NLP and DBpedia Workshop in conjunction with the 12th International Semantic Web Conference (ISWC 2013) Sydney, Australia, October, 2013.*, volume 1064 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2013.

[13] Sebastian Hellmann, Agata Filipowska, Caroline Barriere, Pablo N. Mendes, and Dimitris Kontokostas. NLP & DBpedia—An Upward Knowledge Acquisition Spiral. In *Proceedings of 1st International Workshop on NLP and DBpedia, October 21-25, Sydney, Australia*, volume 1064 of *NLP & DBpedia 2013*, Sydney, Australia, 10 2013. CEUR Workshop Proceedings.

[14] Magnus Knuth, Dimitris Kontokostas, and Harald Sack, editors. *Proceedings of the 1st Workshop on Linked Data Quality (LDQ)*, number 1215 in CEUR Workshop Proceedings, Aachen, 2014.

[15] Dimitris Kontokostas, Charalampos Bratsas, Sören Auer, Sebastian Hellmann, Ioannis Antoniou, and George Metakides.

kontokostas@informatik.uni-leipzig.de
http://aksw.org/DimitrisKontokostas
June 06, 1981 in Veria, Greece

Towards linked data internationalization - realizing the greek dbpedia. In *Proceedings of the ACM WebSci'11*, 2011.

[16] Dimitris Kontokostas, Charalampos Bratsas, Sören Auer, Sebastian Hellmann, Ioannis Antoniou, and George Metakides. Internationalization of linked data: The case of the greek dbpedia edition. *Web Semantics: Science, Services and Agents on the World Wide Web*, 15(0):51 – 61, 2012.

[17] Dimitris Kontokostas, Martin Brümmer, Sebastian Hellmann, Jens Lehmann, and Lazaros Ioannidis. Nlp data cleansing based on linguistic ontology constraints. In *Proc. of the Extended Semantic Web Conference 2014*, 2014.

[18] Dimitris Kontokostas and Sebastian Hellmann. The DBpedia data stack—towards a sustainable DBpedia project to provide a public data infrastructure for europe. In *EDF2014 Poster Session*, 2014.

[19] Dimitris Kontokostas, Christian Mader, Christian Dirschl, Katja Eck, Michael Leuthold, Jens Lehmann, and Sebastian Hellmann. Semantically enhanced quality assurance in the jurion business use case. In *13th International Conference, ESWC 2016, Heraklion, Crete, Greece, May 2016*, pages 661–676, 2016.

[20] Dimitris Kontokostas, Patrick Westphal, Sören Auer, Sebastian Hellmann, Jens Lehmann, and Roland Cornelissen. 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, pages 115–118. International World Wide Web Conferences Steering Committee, 2014.

[21] Dimitris Kontokostas, Patrick Westphal, Sören Auer, Sebastian Hellmann, Jens Lehmann, Roland Cornelissen, and Amrapali Zaveri. Test-driven evaluation of linked data quality. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 747–758. International World Wide Web Conferences Steering Committee, 2014.

[22] Dimitris Kontokostas, Amrapali Zaveri, Sören Auer, and Jens Lehmann. Triplecheckmate: A tool for crowdsourcing the quality

kontokostas@informatik.uni-leipzig.de
http://aksw.org/DimitrisKontokostas
June 06, 1981 in Veria, Greece

assessment of linked data. In *Proceedings of the 4th Conference on Knowledge Engineering and Semantic Web*, 2013.

[23] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 6(2):167–195, 2015.

[24] Denis Lukovnikov, Claus Stadler, Dimitris Kontokostas, Sebastian Hellmann, and Jens Lehmann. Dbpedia viewer - an integrative interface for dbpedia leveraging the dbpedia service eco system. In *Proc. of the Linked Data on the Web 2014 Workshop*, 2014.

[25] Ciro Baron Neto, Dimitris Kontokostas, Sebastian Hellmann, Kay Müller, and Martin Brümmer. Assessing quantity and quality of links between linked data datasets. In *Proceedings of the Workshop on Linked Data on the Web co-located with the 25th International World Wide Web Conference (WWW 2016)*, apr 2016.

[26] Ciro Baron Neto, Dimitris Kontokostas, Sebastian Hellmann, Kay Müller, and Martin Brümmer. Lodvader: An interface to lod visualization, analytics and discovery in real-time. In *25th WWW conference*, 2016.

[27] Ciro Baron Neto, Dimitris Kontokostas, Gustavo Publio, Kay Müller, Sebastian Hellmann, and Eduardo Moletta. Ld-lex: Linked dataset link extractor. In *ODBASE 2016 - The 15th International Conference on Ontologies, DataBases, and Applications of Semantics*. Springer International Publishing, oct 2016.

[28] Anisa Rula, Amrapali Zaveri, Magnus Knuth, and Dimitris Kontokostas, editors. *Proceedings of the 2nd Workshop on Linked Data Quality (LDQ)*, number 1376 in CEUR Workshop Proceedings, Portorož, 2015.

[29] Gaurav Vaidya, Dimitris Kontokostas, Magnus Knuth, Jens Lehmann, and Sebastian Hellmann. DBpedia Commons: Structured Multimedia Metadata from the Wikimedia Commons. In

Proceedings of the 14th International Semantic Web Conference, October 2015.

[30] Andre Valdestilhas, Natanael Arndt, and Dimitris Kontokostas. Dbpediasameas: An approach to tackle heterogeneity in dbpedia identifiers. In *SEMANTiCS 2015*, 2015.

[31] Amrapali Zaveri, Dimitris Kontokostas, Mohamed Ahmed Sherif, Lorenz Bühmann, Mohamed Morsey, Sören Auer, and Jens Lehmann. User-driven quality evaluation of dbpedia. In *Proceedings of 9th International Conference on Semantic Systems, I-SEMANTICS '13, Graz, Austria, September 4-6, 2013*, pages 97–104. ACM, 2013.

kontokostas@informatik.uni-leipzig.de
http://aksw.org/DimitrisKontokostas
June 06, 1981 in Veria, Greece

SELBSTÄNDIGKEITSERKLÄRUNG

Hiermit erkläre ich, die vorliegende Dissertation selbständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialien oder erbrachten Dienstleistungen als solche gekennzeichnet.

Leipzig, den 22.3.2017

Dimitrios Kontokostas