Introduction to Linked Data and its Lifecycle on the Web

Sören Auer, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Amrapali Zaveri

AKSW, Institut für Informatik, Universität Leipzig, Pf 100920, 04009 Leipzig
{lastname}@informatik.uni-leipzig.de
http://aksw.org

Abstract. With Linked Data, a very pragmatic approach towards achieving the vision of the Semantic Web has gained some traction in the last years. The term Linked Data refers to a set of best practices for publishing and interlinking structured data on the Web. While many standards, methods and technologies developed within by the Semantic Web community are applicable for Linked Data, there are also a number of specific characteristics of Linked Data, which have to be considered. In this article we introduce the main concepts of Linked Data. We present an overview of the Linked Data lifecycle and discuss individual approaches as well as the state-of-the-art with regard to extraction, authoring, linking, enrichment as well as quality of Linked Data. We conclude the chapter with a discussion of issues, limitations and further research and development challenges of Linked Data. This article is an updated version of a similar lecture given at Reasoning Web Summer School 2011.

1 Introduction

One of the biggest challenges in the area of intelligent information management is the exploitation of the Web as a platform for data and information integration as well as for search and querying. Just as we publish unstructured textual information on the Web as HTML pages and search such information by using keyword-based search engines, we are already able to easily publish structured information, reliably interlink this information with other data published on the Web and search the resulting data space by using more expressive querying beyond simple keyword searches.

The Linked Data paradigm has evolved as a powerful enabler for the transition of the current document-oriented Web into a Web of interlinked Data and, ultimately, into the Semantic Web. The term Linked Data here refers to a set of best practices for publishing and connecting structured data on the Web. These best practices have been adopted by an increasing number of data providers over the past three years, leading to the creation of a global data space that contains many billions of assertions – the Web of Linked Data (cf. Figure 1).

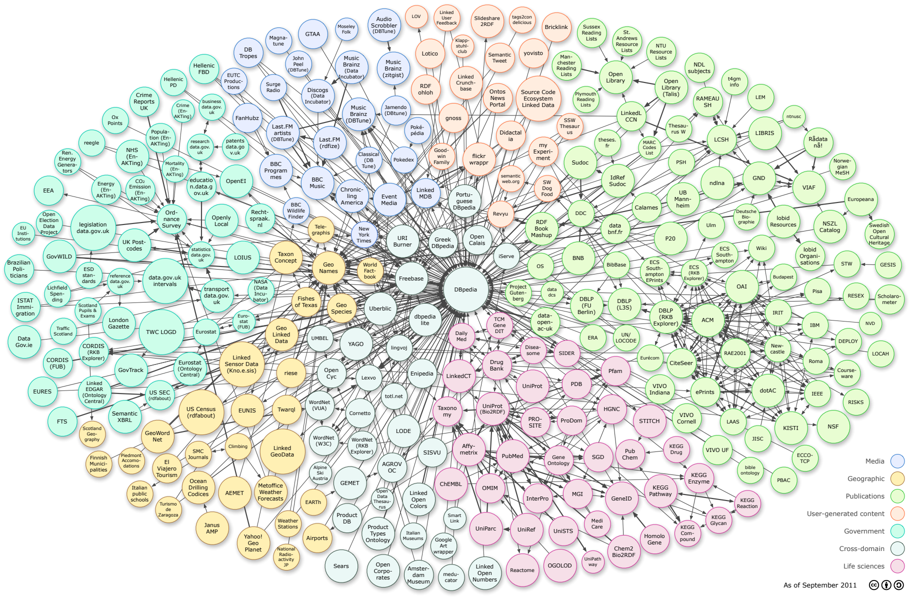


Fig. 1: Overview of some of the main Linked Data knowledge bases and their interlinks available on the Web. (This overview is published regularly at `http://lod-cloud.net` and generated from the Linked Data packages described at the dataset metadata repository `ckan.net`.)

In this chapter we give an overview of recent development in the area of Linked Data management. The different stages in the linked data life-cycle [11,7] are depicted in Figure 2.

Information represented in unstructured form or adhering to other structured or semi-structured representation formalisms must be mapped to the RDF data model (*Extraction*). Once there is a critical mass of RDF data, mechanisms have to be in place to store, index and query this RDF data efficiently (*Storage & Querying*). Users must have the opportunity to create new structured information or to correct and extend existing ones (*Authoring*). If different data publishers provide information about the same or related entities, links between those different information assets have to be established (*Linking*). Since Linked Data primarily comprises instance data we observe a lack of classification, structure and schema information. This deficiency can be tackled by approaches for enriching data with higher-level structures in order to be able to aggregate and query the data more efficiently (*Enrichment*). As with the Document Web, the Data Web contains a variety of information of different quality. Hence, it is important to devise strategies for assessing the quality of data published on the Data Web (*Quality Analysis*). Once problems are detected, strategies for repairing these problems and supporting the evolution of Linked Data are required (*Evolution & Repair*). Last but not least, users have to be empowered to browse, search and explore the structure information available on the Data Web in a fast and user friendly manner (*Search, Browsing & Exploration*).

These different stages of the linked data life-cycle do not exist in isolation or are passed in a strict sequence, but mutually fertilize themselves. Examples include the following:

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

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

The use of Linked Data offers a number of significant benefits:

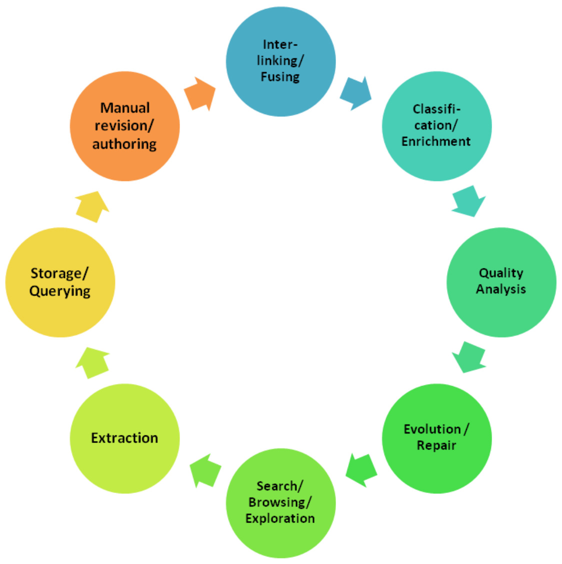

Inter-linking/ Fusing
Classifi-cation/ Enrichment
Quality Analysis
Evolution / Repair
Search/ Browsing/ Exploration
Extraction
Storage/ Querying
Manual revision/ authoring

Fig. 2: The Linked Data life-cycle.

– *Uniformity.* All datasets published as Linked Data share a uniform data model, the RDF statement data model. With this data model all information is represented in facts expressed as triples consisting of a subject, predicate and object. The elements used in subject, predicate or object positions are mainly globally unique IRI/URI entity identifiers. At the object position also literals, i.e. typed data values can be used.
– *De-referencability.* URIs are not just used for identifying entities, but since they can be used in the same way as URLs they also enable locating and retrieving resources describing and representing these entities on the Web.
– *Coherence.* When an RDF triple contains URIs from different namespaces in subject and object position, this triple basically establishes a link between the entity identified by the subject (and described in the source dataset using namspace A) with the entity identified by the object (described in the target dataset using namespace B). Through the typed RDF links, data items are effectively interlinked.
– *Integrability.* Since all Linked Data sources share the RDF data model, which is based on a single mechanism for representing information, it is very easy to attain a syntactic and simple semantic integration of different Linked Data sets. A higher level semantic integration can be achieved by employing schema and instance matching techniques and expressing found matches again as alignments of RDF vocabularies and ontologies in terms of additional triple facts.
– *Timeliness.* Publishing and updating Linked Data is relatively simple thus facilitating a timely availability. In addition, once a Linked Data source is updated it is straightforward to access and use the updated data source, since time consuming and error prune extraction, transformation and loading is not required.

Representation \ degree of openness	Possibly closed	Open (cf. `opendefinition.org`)
Structured data model (i.e. XML, CSV, SQL etc.)	**Data**	**Open Data**
RDF data model (published as Linked Data)	**Linked Data** (LD)	**Linked Open Data** (LOD)

Table 1: Juxtaposition of the concepts Linked Data, Linked Open Data and Open Data.

The development of research approaches, standards, technology and tools for supporting the Linked Data lifecycle data is one of the main challenges. Developing adequate and pragmatic solutions to these problems can have a substantial impact on science, economy, culture and society in general. The publishing, integration and aggregation of statistical and economic data, for example, can help to obtain a more precise and timely picture of the state of our economy. In the domain of health care and life sciences making sense of the wealth of structured information already available on the Web can help to improve medical information systems and thus make health care more adequate and efficient. For the media and news industry, using structured background information from the Data Web for enriching and repurposing the quality content can facilitate the creation of new publishing products and services. Linked Data technologies can help to increase the flexibility, adaptability and efficiency of information management in organizations, be it companies, governments and public administrations or online communities. For end-users and society in general, the Data Web will help to obtain and integrate required information more efficiently and thus successfully manage the transition towards a knowledge-based economy and an information society.

Structure of this chapter. This chapter aims to explain the foundations of Linked Data and introducing the different aspects of the Linked Data lifecycle by highlighting a particular approach and providing references to related work and further reading. We start by briefly explaining the principles underlying the Linked Data paradigm in Section 2. The first aspect of the Linked Data lifecycle is the extraction of information from unstructured, semi-structured and structured sources and their representation according to the RDF data model (Section 3). We present the user friendly authoring and manual revision aspect of Linked Data with the example of Semantic Wikis in Section 4. The interlinking aspect is tackled in Section 5 and gives an overview on the LIMES framework. We describe how the instance data published and commonly found on the Data Web can be enriched with higher level structures in Section 6. We present an overview of the various data quality dimensions and metrics along with currently existing tools for data quality assessment of Linked Data in Section 7. Due to space limitations we omit a detailed discussion of the evolution as well as search, browsing and exploration aspects of the Linked Data lifecycle in this chapter. The chapter is concluded by several sections on promising applications of Linked Data and semantic technologies, in particular Open Governmental Data, Semantic Business Intelligence and Statistical and Economic Data.

Overall, this is an updated version of a similar lecture given at Reasoning Web Summer School 2011 [13].

2 The Linked Data paradigm

In this section we introduce the basic principles of Linked Data. The section is partially based on the Section 2 from [53]. The term Linked Data refers to a set of best practices for publishing and interlinking structured data on the Web. These best practices were introduced by Tim Berners-Lee in his Web architecture note Linked Data[1] and have become known as the Linked Data principles. These principles are:

- Use URIs as names for things.
- Use HTTP URIs so that people can look up those names.
- When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL).
- Include links to other URIs, so that they can discover more things.

The basic idea of Linked Data is to apply the general architecture of the World Wide Web [70] to the task of sharing structured data on global scale. The Document Web is built on the idea of setting hyperlinks between Web documents that may reside on different Web servers. It is built on a small set of simple standards: Uniform Resource Identifiers (URIs) and their extension Internationalized Resource Identifiers (IRIs) as globally unique identification mechanism [21], the Hypertext Transfer Protocol (HTTP) as universal access mechanism [43], and the Hypertext Markup Language (HTML) as a widely used content format [64]. Linked Data builds directly on Web architecture and applies this architecture to the task of sharing data on global scale.

2.1 Resource identification with IRIs

To publish data on the Web, the data items in a domain of interest must first be identified. These are the things whose properties and relationships will be described in the data, and may include Web documents as well as real-world entities and abstract concepts. As Linked Data builds directly on Web architecture, the Web architecture term *resource* is used to refer to these *things of interest*, which are in turn identified by HTTP URIs. Linked Data uses only HTTP URIs, avoiding other URI schemes such as URNs [100] and DOIs[2]. The structure of HTTP URIs looks as follows:

```
[scheme:][//authority][path][?query][#fragment]
```

A URI for identifying Shakespeare's 'Othello', for example, could look as follows:

```
http://de.wikipedia.org/wiki/Othello#id
```

HTTP URIs make good names for two reasons:

1. They provide a simple way to create globally unique names in a decentralized fashion, as every owner of a domain name or delegate of the domain name owner may create new URI references.
2. They serve not just as a name but also as a means of accessing information describing the identified entity.

[1] `http://www.w3.org/DesignIssues/LinkedData.html.`
[2] `http://www.doi.org/hb.html`

2.2 De-referencability

Any HTTP URI should be de-referencable, meaning that HTTP clients can look up the URI using the HTTP protocol and retrieve a description of the resource that is identified by the URI. This applies to URIs that are used to identify classic HTML documents, as well as URIs that are used in the Linked Data context to identify real-world objects and abstract concepts. Descriptions of resources are embodied in the form of Web documents. Descriptions that are intended to be read by humans are often represented as HTML. Descriptions that are intended for consumption by machines are represented as RDF data. Where URIs identify real-world objects, it is essential to not confuse the objects themselves with the Web documents that describe them. It is therefore common practice to use different URIs to identify the real-world object and the document that describes it, in order to be unambiguous. This practice allows separate statements to be made about an object and about a document that describes that object. For example, the creation year of a painting may be rather different to the creation year of an article about this painting. Being able to distinguish the two through use of different URIs is critical to the consistency of the Web of Data.

The Web is intended to be an information space that may be used by humans as well as by machines. Both should be able to retrieve representations of resources in a form that meets their needs, such as HTML for humans and RDF for machines. This can be achieved using an HTTP mechanism called content negotiation [43]. The basic idea of content negotiation is that HTTP clients send HTTP headers with each request to indicate what kinds of documents they prefer. Servers can inspect these headers and select an appropriate response. If the headers indicate that the client prefers HTML then the server will respond by sending an HTML document If the client prefers RDF, then the server will send the client an RDF document.

There are two different strategies to make URIs that identify real-world objects de-referencable [136]. Both strategies ensure that objects and the documents that describe them are not confused and that humans as well as machines can retrieve appropriate representations.

303 URIs. Real-world objects can not be transmitted over the wire using the HTTP protocol. Thus, it is also not possible to directly de-reference URIs that identify real-world objects. Therefore, in the 303 URI strategy, instead of sending the object itself over the network, the server responds to the client with the HTTP response code `303 See Other` and the URI of a Web document which describes the real-world object. This is called a *303 redirect*. In a second step, the client de-references this new URI and retrieves a Web document describing the real-world object.

Hash URIs. A widespread criticism of the 303 URI strategy is that it requires two HTTP requests to retrieve a single description of a real-world object. One option for avoiding these two requests is provided by the hash URI strategy. The hash URI strategy builds on the characteristic that URIs may contain a special part that is separated from the base part of the URI by a hash symbol (#). This special part is called the fragment identifier. When a client wants to retrieve a hash URI the HTTP protocol requires the fragment part to be stripped off before requesting the URI from the server. This means a URI that includes a hash cannot be retrieved directly, and therefore does not necessarily identify a Web document. This enables such URIs to be used to identify real-world objects and abstract concepts, without creating ambiguity [136].

Both approaches have their advantages and disadvantages. Section 4.4. of the W3C Interest Group Note *Cool URIs for the Semantic Web* compares the two approaches [136]: Hash URIs have the advantage of reducing the number of necessary HTTP round-trips, which in turn reduces access latency. The downside of the hash URI approach is that the descriptions of all resources that share the same non-fragment URI part are always returned to the client together, irrespective of whether the client is interested in only one URI or all. If these descriptions consist of a large number of triples, the hash URI approach can lead to large amounts of data being unnecessarily transmitted to the client. 303 URIs, on the other hand, are very flexible because the redirection target can be configured separately for each resource. There could be one describing document for each resource, or one large document for all of them, or any combination in between. It is also possible to change the policy later on.

2.3 RDF Data Model

The RDF data model [1] represents information as sets of statements, which can be visualized as node-and-arc-labeled directed graphs. The data model is designed for the integrated representation of information that originates from multiple sources, is heterogeneously structured, and is represented using different schemata. RDF can be viewed as a *lingua franca*, capable of moderating between other data models that are used on the Web.

In RDF, information is represented in statements, called RDF triples. The three parts of each triple are called its subject, predicate, and object. A triple mimics the basic structure of a simple sentence, such as for example:

```
Burkhard Jung   is the mayor of   Leipzig
  (subject)       (predicate)     (object)
```

The following is the formal definition of RDF triples as it can be found in the W3C RDF standard [1].

Definition 1 (RDF Triple). *Assume there are pairwise disjoint infinite sets I, B, and L representing IRIs, blank nodes, and RDF literals, respectively. A triple $(v_1, v_2, v_3) \in (I \cup B) \times I \times (I \cup B \cup L)$ is called an RDF triple. In this tuple, v_1 is the subject, v_2 the predicate and v_3 the object. We denote the union $I \cup B \cup L$ by T called RDF terms.*

The main idea is to use IRIs as identifiers for entities in the subject, predicate and object positions in a triple. Data values can be represented in the object position as literals. Furthermore, the RDF data model also allows in subject and object positions the use of identifiers for unnamed entities (called blank nodes), which are not globally unique and can thus only be referenced locally. However, the use of blank nodes is discouraged in the Linked Data context as we discuss below. Our example fact sentence about Leipzig's mayor would now look as follows:

```
<http://leipzig.de/id>
                <http://example.org/p/hasMayor>
                                            <http://Burkhard-Jung.de/id> .
      (subject)           (predicate)                (object)
```

This example shows, that IRIs used within a triple can originate from different namespaces thus effectively facilitating the mixing and mashing of different RDF vocabularies and entities from different Linked Data knowledge bases. A triple having identifiers from different knowledge bases at subject and object position can be also viewed as an typed link between the entities identified by subject and object. The predicate then identifies the type of link. If we combine different triples we obtain an RDF graph.

Definition 2 (RDF Graph). *A finite set of RDF triples is called RDF graph. The RDF graph itself represents an resource, which is located at a certain location on the Web and thus has an associated IRI, the graph IRI.*

An example of an RDF graph is depicted in Figure 3. Each unique subject or object contained in the graph is visualized as a node (i.e. oval for resources and rectangle for literals). Predicates are visualized as labeled arcs connecting the respective nodes. There are a number of synonyms being used for RDF graphs, all meaning essentially the same but stressing different aspects of an RDF graph, such as *RDF document* (file perspective), *knowledge base* (collection of facts), *vocabulary* (shared terminology), *ontology* (shared logical conceptualization).

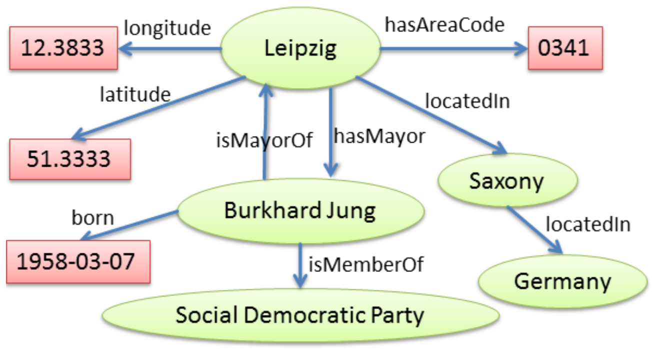


Fig. 3: Example RDF graph containing 9 triples describing the city of Leipzig and its mayor.

Problematic RDF features in the Linked Data Context Besides the features mentioned above, the RDF Recommendation [1] also specifies some other features. In order to make it easier for clients to consume data only the subset of the RDF data model described above should be used. In particular, the following features are problematic when publishing RDF as Linked Data:

- *RDF reification* (for making statements about statements) should be avoided if possible, as reified statements are rather cumbersome to query with the SPARQL query language. In many cases using reification to publish metadata about individual RDF statements can be avoided by attaching the respective metadata to the RDF document containing the relevant triples.
- *RDF collections* and *RDF containers* are also problematic if the data needs to be queried with SPARQL. Therefore, in cases where the relative ordering of items in a set is not significant, the use of multiple triples with the same predicate is recommended.
- The scope of *blank nodes* is limited to the document in which they appear, meaning it is not possible to create links to them from external documents. In addition, it is more difficult to merge data from different sources when blank nodes are used, as there is no URI to serve as a common key. Therefore, all resources in a data set should be named using IRI references.

2.4 RDF serializations

The initial official W3C RDF standard [1] comprised a serialization of the RDF data model in XML called *RDF/XML*. Its rationale was to integrate RDF with the existing XML standard, so it could be used smoothly in conjunction with the existing XML technology landscape. Unfortunately, RDF/XML turned out to be rather difficult to understand for the majority of potential users, since it requires to be familiar with two data models (i.e. the tree-oriented XML data model as well as the statement oriented RDF datamodel) and interactions between them, since RDF statements are represented in XML. As a consequence, with *N-Triples*, *Turtle* and *N3* a family of alternative text-based RDF serializations was developed, whose members have the same origin, but balance differently between readability for humans and machines. Later in 2009, *RDFa* (RDF Annotations, [2]) was standardized by the W3C in order to simplify the integration of HTML and RDF and to allow the joint representation of structured and unstructured content within a single source HTML document. Another RDF serialization, which is particularly beneficial in the context of JavaScript web applications and mashups is the serialization of RDF in JSON. In the sequel we present each of these RDF serializations in some more detail. Figure 5 presents an example serialized in the most popular serializations.

N-Triples. This serialization format was developed specifically for RDF graphs. The goal was to create a serialization format which is very simple. N-Triples are easy to parse and generate by software. An N-Triples document consists of a set of triples, which are separated '.' (lines 1-2, 3-4 and 5-6 in Figure 5 contain one triple each). URI components of a triple are written in full and enclosed by '<' and '>'. Literals are enclosed in quotes, datatypes can be appended to a literal using 'g (line 6), language tags using '@' (line 4). They are a subset of *Notation 3* and *Turtle* but lack, for example, shortcuts such as CURIEs. This makes them less readable and more difficult to create manually. Another disadvantage is that N-triples use only the 7-bit US-ASCII character encoding instead of UTF-8.

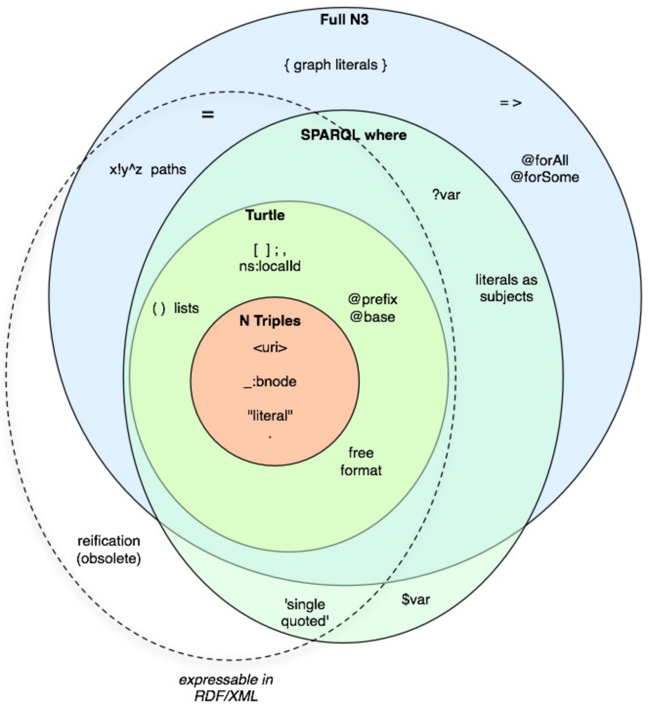


Fig. 4: Various textual RDF serializations as subsets of N3 (from [20]).

Turtle. Turtle (Terse RDF Triple Language) is a subset of, and compatible with, Notation 3 and a superset of the minimal N-Triples format (cf. Figure 4). The goal was to use the essential parts of Notation 3 for the serialization of RDF models and omit everything else. Turtle became part of the SPARQL query language for expressing graph patterns. Compared to N-Triples, Turtle introduces a number of shortcuts, such as namespace definitions (lines 1-5 in Figure 5), the semicolon as a separator between triples sharing the same subject (which then does not have to be repeated in subsequent triples) and the comma as a separator between triples sharing the same subject and predicate. Turtle, just like Notation 3, is human-readable, and can handle the "%" character in URIs (required for encoding special characters) as well as IRIs due to its UTF-8 encoding.

Notation 3. N3 (Notation 3) was devised by Tim Berners-Lee and developed for the purpose of serializing RDF. The main aim was to create a very human-readable serialization. Hence, an RDF model serialized in N3 is much more compact than the same model in RDF/XML but still allows a great deal of expressiveness even going beyond the RDF data model in some aspects. Since, the encoding for N3 files is UTF-8 the use of IRIs does not pose a problem.

RDF/XML. The RDF/XML syntax [97] is standardized by the W3C and is widely used to publish Linked Data on the Web. However, the syntax is also viewed as difficult for humans to read and write, and therefore consideration should be given to using other serializations in data management and curation workflows that involve human intervention, and to the provision of alternative serializations for consumers who may wish to eyeball the data. The MIME type that should be used for RDF/XML within HTTP content negotiation is `application/rdf+xml`.

RDFa. RDF in Attributes (RDFa, [2]) was developed for embedding RDF into XHTML pages. Since it is an extension to the XML based XHTML, UTF-8 and UTF-16 are used for encoding. The "%" character for URIs in triples can be used because RDFa tags are not used for a part of a RDF statement. Thus IRIs are usable, too. Because RDFa is embedded in XHTML, the overhead is higher compared to other serialization technologies and also reduces the readability. The basic idea of RDFa is enable an RDFa processor to extract RDF statements from an RDFa enriched HTML document. This is achieved by defining the scope of a certain resource description, for example, using the 'about' attribute (cf. line 10 in Figure 5). Within this scope, triples can now be extracted from links having an additional 'rel' attribute (line 13) or other tags having a 'property attribute' (lines 11 and 14).

JSON-LD. JavaScript Object Notation (JSON) was developed for easy data interchange between applications. JSON, although carrying JavaScript in its name and being a subset of JavaScript, meanwhile became a language independent format which can be used for exchanging all kinds of data structures and is widely supported in different programming languages. Compared to XML, JSON-LD requires less overhead with regard to parsing and serializing. JSON-LD has been developed by the JSON for Linking Data Community Group and been transferred to the RDF Working Group for review, improvement, and publication along the Recommendation track. JSON-LD's design goals are simplicity, compatibility, expressiveness, terseness, zero edits and one-pass processing. As a result, JSON-LD documents are basically standard attribute-value JSON documents with an additional context section (lines 2-7 in Figure 5) establishing mappings to RDF vocabularies. Text in JSON and, thus, also RDF resource identifiers are encoded in Unicode and hence can contain IRIs.

N-Triples

```
<http://dbpedia.org/resource/Leipzig> <http://dbpedia.org/property/hasMayor>
      <http://dbpedia.org/resource/Burkhard_Jung> .
<http://dbpedia.org/resource/Leipzig> <http://www.w3.org/2000/01/rdf-schema#label>
      "Leipzig"@de .
<http://dbpedia.org/resource/Leipzig> <http://www.w3.org/2003/01/geo/wgs84_pos#lat>
      "51.333332"^^<http://www.w3.org/2001/XMLSchema#float> .
```

Turtle

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs="http://www.w3.org/2000/01/rdf-schema#> .
@prefix dbp="http://dbpedia.org/resource/> .
@prefix dbpp="http://dbpedia.org/property/> .
@prefix geo="http://www.w3.org/2003/01/geo/wgs84_pos#> .

dbp:Leipzig  dbpp:hasMayor  dbp:Burkhard_Jung ;
             rdfs:label     "Leipzig"@de ;
             geo:lat        "51.333332"^^xsd:float .
```

RDF/XML

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:dbpp="http://dbpedia.org/property/"
         xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#">
 <rdf:Description rdf:about="http://dbpedia.org/resource/Leipzig">
  <property:hasMayor rdf:resource="http://dbpedia.org/resource/Burkhard_Jung" />
  <rdfs:label xml:lang="de">Leipzig</rdfs:label>
  <geo:lat rdf:datatype="http://www.w3.org/2001/XMLSchema#float">51.3333</geo:lat>
 </rdf:Description>
</rdf:RDF>
```

RDFa

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
    "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
<html version="XHTML+RDFa 1.0" xml:lang="en" xmlns="http://www.w3.org/1999/xhtml"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:dbpp="http://dbpedia.org/property/"
      xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#">
  <head><title>Leipzig</title></head>
  <body about="http://dbpedia.org/resource/Leipzig">
    <h1 property="rdfs:label" xml:lang="de">Leipzig</h1>
    <p>Leipzig is a city in Germany. Leipzig's mayor is
     <a href="Burkhard_Jung" rel="dbpp:hasMayor">Burkhard Jung</a>. It is located
     at latitude <span property="geo:lat" datatype="xsd:float">51.3333</span>.</p>
  </body>
</html>
```

JSON-LD

```
{
 "@context": {
   "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
   "hasMayor": { "@id": "http://dbpedia.org/property/hasMayor", "@type": "@id" },
   "Person": "http://xmlns.com/foaf/0.1/Person",
   "lat": "http://www.w3.org/2003/01/geo/wgs84_pos#lat"
 },
 "@id": "http://dbpedia.org/resource/Leipzig",
 "rdfs:label": "Leipzig",
 "hasMayor": "http://dbpedia.org/resource/Burkhard_Jung",
 "lat": { "@value": "51.3333", "@type": "http://www.w3.org/2001/XMLSchema#float"
}
```

Fig. 5: Different RDF serializations of three triples from Figure 3.

3 Extraction

Information represented in unstructured form or adhering to a different structured representation formalism must be mapped to the RDF data model in order to be used within the Linked Data life-cycle. In this section, we give an overview on some relevant approaches for extracting RDF from unstructured and structured sources.

3.1 From Unstructured Sources

The extraction of structured information from unstructured data sources (especially text) has been a central pillar of *natural language processing* (NLP) and *Information Extraction* (IE) for several decades. With respect to the extraction of RDF data from unstructured data, three sub-disciplines of NLP play a central role: *Named Entity Recognition* (NER) for the extraction of entity labels from text, *Keyword/Keyphrase Extraction* (KE) for the recognition of central topics and *Relationship Extraction* (RE, also called relation mining) for mining the properties which link the entities and keywords described in the data source. A noticeable additional task during the migration of these techniques to Linked Data is the extraction of suitable IRIs for the discovered entities and relations, a requirement that was not needed before. In this section, we give a short overview of approaches that implement the required NLP functionality. Then we present a framework that applies machine learning to boost the quality of the RDF extraction from unstructured data by merging the results of NLP tools. As an orthogonal activity, we want to mention the NLP2RDF project [57], which provides RDF serialisation for NLP tools solving the above mentioned tasks.

Named Entity Recognition. The goal of NER is to discover instances of a predefined classes of entities (e.g., persons, locations, organizations) in text. NER tools and frameworks implement a broad spectrum of approaches, which can be subdivided into three main categories: dictionary-based, rule-based, and machine-learning approaches. The first systems for NER implemented dictionary-based approaches, which relied on a list of NEs and tried to identify these in text [156,5]. Following work that showed that these approaches did not perform well for NER tasks such as recognizing proper names [135], rule-based approaches were introduced. These approaches rely on hand-crafted rules [32,145] to recognize NEs. Most rule-based approaches combine dictionary and rule-based algorithms to extend the list of known entities. Nowadays, handcrafted rules for recognizing NEs are usually implemented when no training examples are available for the domain or language to process [105].

When training examples are available, the methods of choice are borrowed from supervised machine learning. Approaches such as Hidden Markov Models [167], Maximum Entropy Models [35] and Conditional Random Fields [44] have been applied to the NER task. Due to scarcity of large training corpora as necessitated by machine learning approaches, semi-supervised [124,104] and unsupervised machine learning approaches [106,40] have also been used for extracting NER from text. [104] gives an exhaustive overview of approaches for NER.

Keyphrase Extraction. Keyphrases/Keywords are multi-word units (MWUs) which capture the main topics of a document. The automatic detection of such MWUs has been an important task of NLP for decades but due to the very ambiguous definition of what an appropriate keyword should be, current approaches to the extraction of keyphrases still display low F-scores [74]. From the point of view of the Semantic Web, the extraction of keyphrases is a very similar task to that of finding tags for a given document. Several categories of approaches have been adapted to enable KE, of which some originate from research areas such as summarization and information retrieval (IR). Still, according to [73], the majority of the approaches to KE implement combinations of statistical, rule-based or heuristic methods [47,119] on mostly document [96], keyphrase [148] or term cohesion features [123]. [74] gives a overview of current tools for KE.

Relation Extraction. The extraction of relations from unstructured data builds upon work for NER and KE to determine the entities between which relations might exist. Most tools for RE rely on pattern-based approaches. Some early work on pattern extraction relied on supervised machine learning [50]. Yet, such approaches demanded large amount of training data, making them difficult to adapt to new relations. The subsequent generation of approaches to RE aimed at bootstrapping patterns based on a small number of input patterns and instances. For example, [28] presents the Dual Iterative Pattern Relation Expansion (DIPRE) and applies it to the detection of relations between authors and titles of books. This approach relies on a small set of seed patterns to maximize the precision of the patterns for a given relation while minimizing their error rate of the same patterns. Snowball [3] extends DIPRE by a new approach to the generation of seed tuples. Newer approaches aim to either collect redundancy information from the whole Web [122] or Wikipedia [157,163] in an unsupervised manner or to use linguistic analysis [52,118] to harvest generic patterns for relations.

URI Disambiguation. One important problem for the integration of NER tools for Linked Data is the retrieval of IRIs for the entities to be manipulated. In most cases, the URIs can be extracted from generic knowledge bases such as DBpedia [103,82] by comparing the label found in the input data with the `rdfs:label` or `dc:title` of the entities found in the knowledge base. Furthermore, information such as the type of NEs can be used to filter the retrieved IRIs via a comparison of the `rdfs:label` of the `rdf:type` of the URIs with the name of class of the NEs. Still in many cases (e.g., Leipzig, Paris), several entities might bear the same label.

Unsupervised Extraction Example: The FOX Framework.

Several frameworks have been developed to implement the functionality above for the Data Web including OpenCalais[3] and Alchemy[4]. Yet, these tools rely mostly on one approach to perform the different tasks at hand. In this section, we present the FOX

[3] http://www.opencalais.com
[4] http://www.alchemyapi.com

(Federated knOwledge eXtraction) framework[5], which makes use of the diversity of the algorithms available for NER, KE and RE to generate high-quality RDF.

The architecture of FOX consists of *three main layers* as shown in Figure 6. The *machine learning* layer implements interfaces for accommodating ensemble learning techniques such as simple veto algorithms but also neural networks. It consists of *two main modules*. The *training module* allows to load training data so as to enable FOX to learn the best combination of tools and categories for achieving superior recall and precision on the input training data. Depending on the training algorithm used, the user can choose to tune the system for either precision or recall. When using neural networks for example, the user can decide to apply a higher threshold for the output neurons, thus improving the precision but potentially limiting the recall. The *prediction module* allows to run FOX by loading the result of a training session and processing the input data according to the tool-category combination learned during the training phase. Note that the same learning approach can by applied to NER, KE, RE and URI lookup as they call all be modelled as classification tasks.

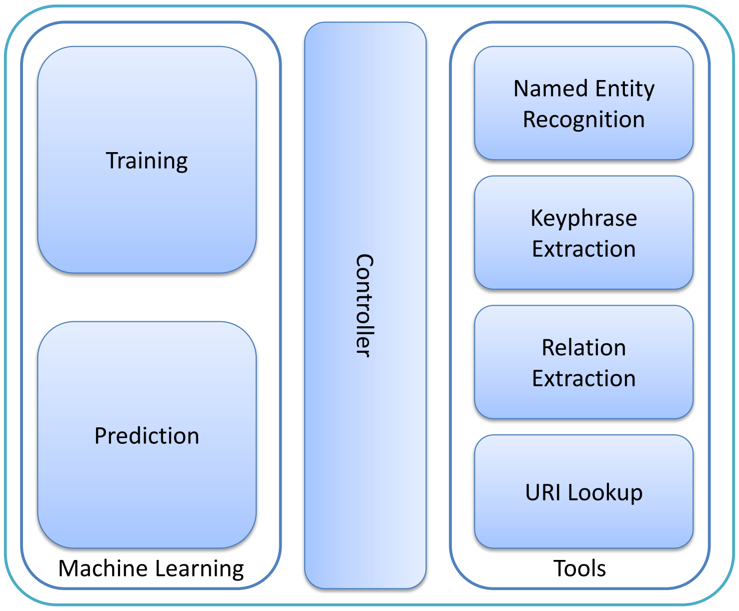


Fig. 6: FOX Architecture

The second layer of FOX is the *controller*, which coordinates the access to the modules that carry out the language processing. The controller is aware of each of the modules in its backend and carries out the initialisation of these modules once FOX is

[5] http://aksw.org/projects/fox

started. Furthermore, it collects the results from the backend modules and invokes the results of a training instance to merge the results of these tools.

The final layer of FOX is the *tool layer*, wherein all NLP tools and services integrated in FOX can be found. It is important to notice that the tools per se are not trained during the learning phase of FOX. Rather, we learn of the models already loaded in the tools to allow for the best prediction of named entities in a given domain.

The ensemble learning implemented by FOX was evaluated in the task of NER by integrating three NER tools (Stanford NER, Illinois NER and a commercial tool) and shown to lead to an improvement of more than 13% in F-Score (see Figure 7) when combining three tools, therewith even outperforming commercial systems.

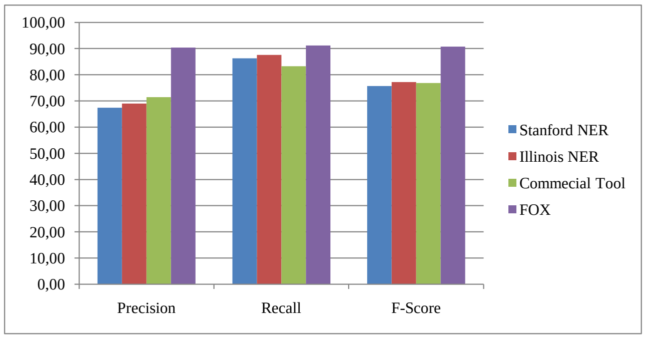


Fig. 7: Comparison of precision, recall and F-score of the best runs of FOX and its components on NER.

3.2 From Structured Sources

Structured knowledge, e.g. relational databases and XML, is the backbone of many (web) applications. Extracting or converting this knowledge to RDF is a long-standing research goal in the Semantic Web community. A conversion to RDF allows to integrate the data with other sources and perform queries over it. In this lecture, we focus on the conversion of relational databases to RDF (see Figure 8). In the first part, we summarize material from a recent relational database to RDF (RDB2RDF) project report. After that, we describe the mapping language R2RML, which is a language for expressing database to RDF conversion mappings. While we focus on relational date, we also want to note that extraction from CSV files is also highly important as illustrated in use cases in the financial [95] and health sector [164,165].

Triplify and RDB2RDF Survey report The table displayed in Figure 9 is taken from the Triplify WWW paper [8]. The survey report [134] furthermore contained a chart(see Figure 10) showing the reference framework for classifying the approaches and an extensive table classifying the approaches (see Figure 11). Another recent survey is [143].

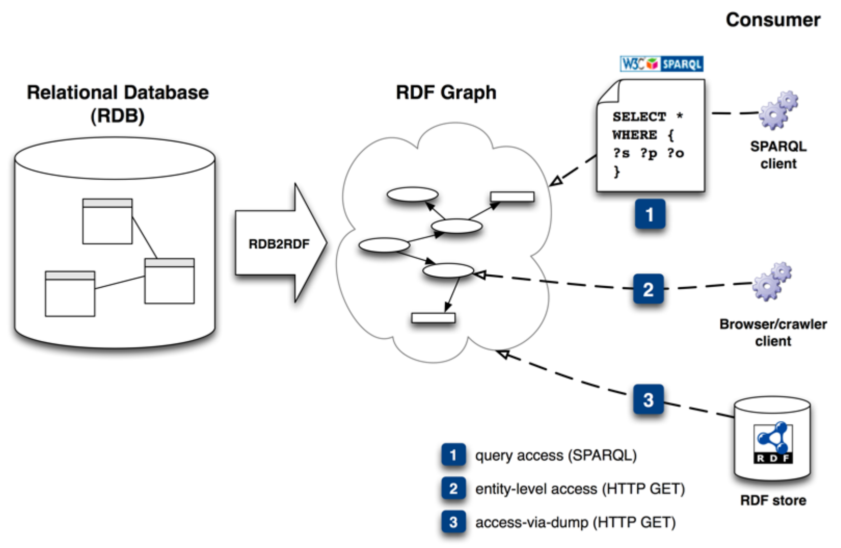
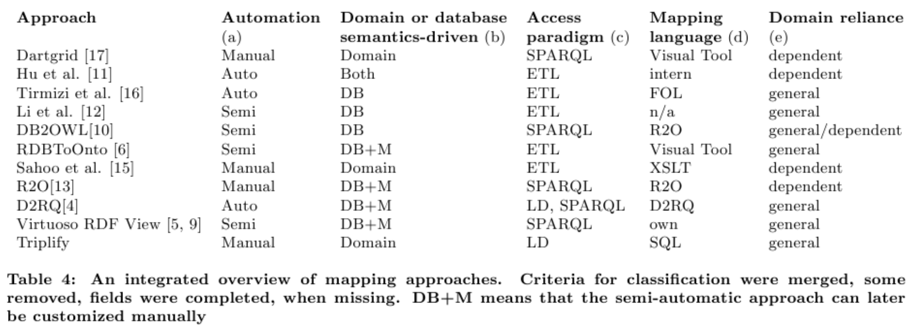

Consumer
Relational Database (RDB)
RDF Graph
RDB2RDF
SELECT * WHERE { ?s ?p ?o }
SPARQL client
Browser/crawler client
RDF store
1 query access (SPARQL)
2 entity-level access (HTTP GET)
3 access-via-dump (HTTP GET)

Fig. 8: Illustration of RDB to RDF conversion.
Source: `http://www.w3.org/2001/sw/rdb2rdf/use-cases/`.

The following criteria can be extracted:

Automation Degree. Degree of mapping creation automation.
Values: Manual, Automatic, Semi-Automatic.

Domain or Database Semantics Driven. Some approaches are tailored to model a domain, sometimes with the help of existing ontologies, while others attempt to extract domain information primarily from the given database schema with few other resources used (domain or database semantics-driven). The latter often results in a table-to-class, column-to-predicate mapping.Some approaches also use a (semi) automatic approach



Approach	**Automation** (a)	**Domain or database semantics-driven** (b)	**Access paradigm** (c)	**Mapping language** (d)	**Domain reliance** (e)
Dartgrid [17]	Manual	Domain	SPARQL	Visual Tool	dependent
Hu et al. [11]	Auto	Both	ETL	intern	dependent
Tirmizi et al. [16]	Auto	DB	ETL	FOL	general
Li et al. [12]	Semi	DB	ETL	n/a	general
DB2OWL[10]	Semi	DB	SPARQL	R2O	general/dependent
RDBToOnto [6]	Semi	DB+M	ETL	Visual Tool	general
Sahoo et al. [15]	Manual	Domain	ETL	XSLT	dependent
R2O[13]	Manual	DB+M	SPARQL	R2O	dependent
D2RQ[4]	Auto	DB+M	LD, SPARQL	D2RQ	general
Virtuoso RDF View [5, 9]	Semi	DB+M	SPARQL	own	general
Triplify	Manual	Domain	LD	SQL	general

Table 4: An integrated overview of mapping approaches. Criteria for classification were merged, some removed, fields were completed, when missing. DB+M means that the semi-automatic approach can later be customized manually

Fig. 9: Table comparing relevant approaches from [8].

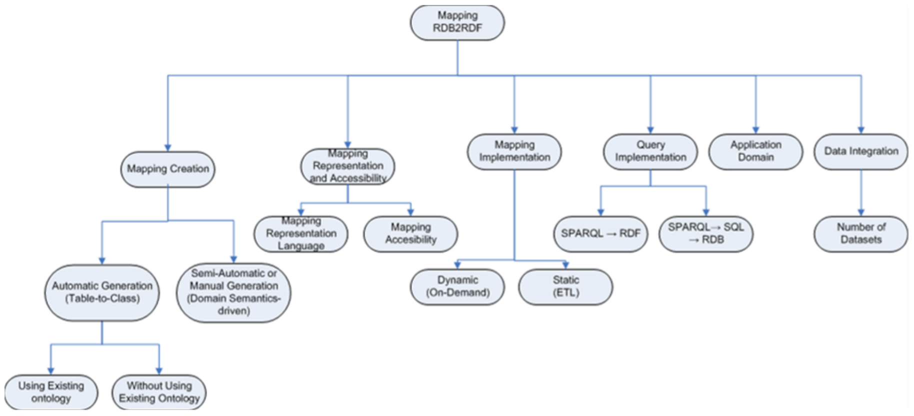


Fig. 10: Reference framework by [134].

based on the database, but allow manual customization to model domain semantics.
Values: Domain, DB (database), DB+M (database and later manual customisation), Both (Domain and DB)

Access Paradigm. Resulting access paradigm (ETL [extract transform load], Linked Data, SPARQL access). Note that the access paradigm also determines whether the resulting RDF model updates automatically. ETL means a one time conversion, while Linked Data and SPARQL always process queries versus the original database.
Values: SPARQL, ETL, LD

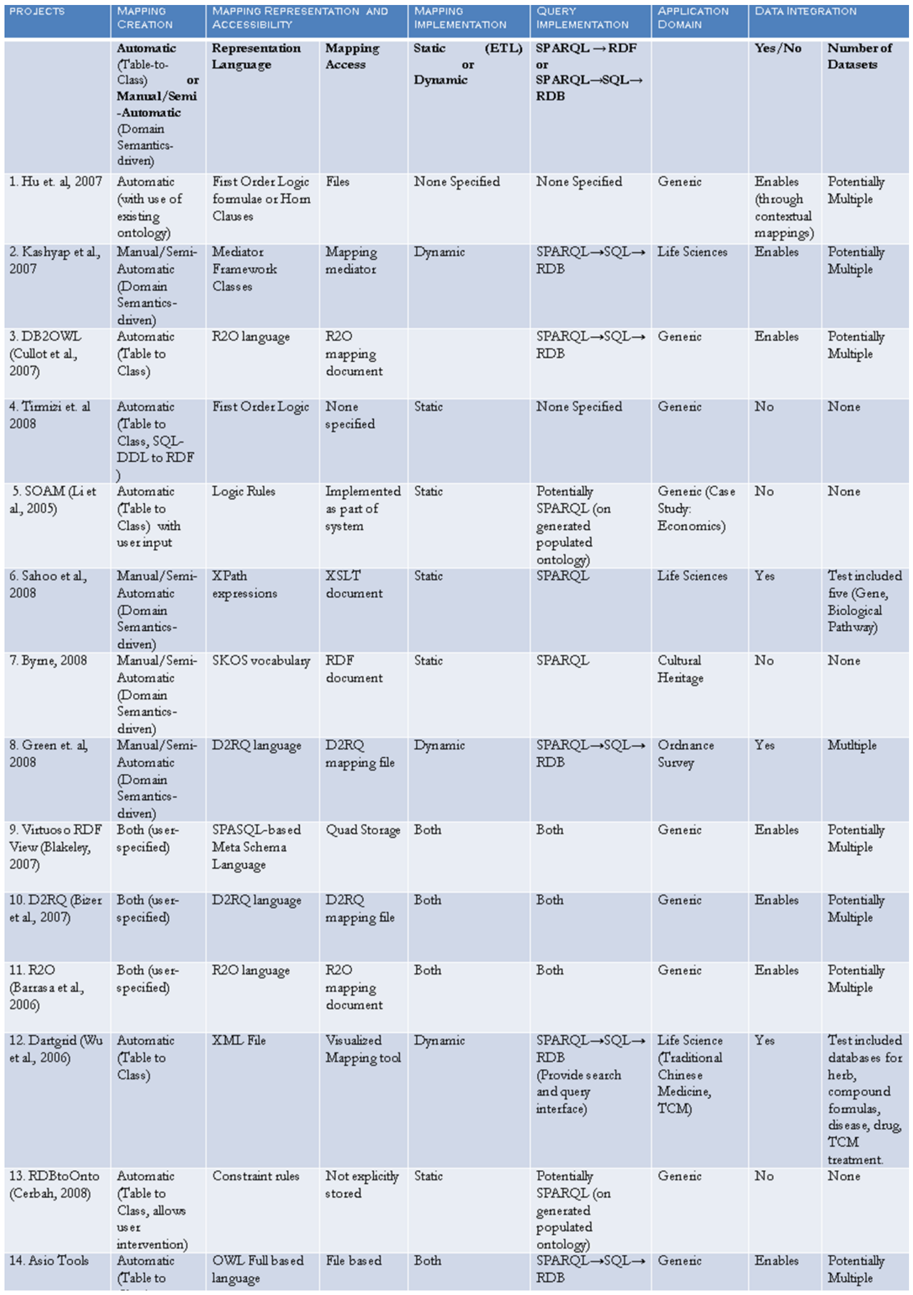
Mapping Language. The used mapping language as an important factor for reusability and initial learning cost.
Values: Visual Tool, intern (internal self-designed language), FOL, n/a (no information available), R2O, XSLT, D2RQ, proprietary, SQL

Domain reliance. Domain reliance (general or domain-dependent): requiring a pre-defined ontology is a clear indicator of domain dependency.
Values: Dependent, General

Type. Although not used in the table the paper discusses four different classes:
Values: Alignment, Database Mining, Integration, Languages/Servers

R2RML - RDB to RDF Mapping Language The R2RML W3C recommendation[6] specifies an RDF notation for mapping relational tables, views or queries into RDF. The primary area of applicability of this is extracting RDF from relational databases, but in special cases R2RML could lend itself to on-the-fly translation of SPARQL into SQL or to converting RDF data to a relational form. The latter application is not the

[6] http://www.w3.org/TR/r2rml/



PROJECTS	MAPPING CREATION	MAPPING REPRESENTATION AND ACCESSIBILITY		MAPPING IMPLEMENTATION	QUERY IMPLEMENTATION	APPLICATION DOMAIN	DATA INTEGRATION	
	Automatic (Table-to-Class) **or Manual/Semi-Automatic** (Domain Semantics-driven)	**Representation Language**	**Mapping Access**	**Static (ETL) or Dynamic**	**SPARQL → RDF or SPARQL→SQL→ RDB**		**Yes/No**	**Number of Datasets**
1. Hu et. al, 2007	Automatic (with use of existing ontology)	First Order Logic formulae or Horn Clauses	Files	None Specified	None Specified	Generic	Enables (through contextual mappings)	Potentially Multiple
2. Kashyap et al., 2007	Manual/Semi-Automatic (Domain Semantics-driven)	Mediator Framework Classes	Mapping mediator	Dynamic	SPARQL→SQL→ RDB	Life Sciences	Enables	Potentially Multiple
3. DB2OWL (Cullot et al., 2007)	Automatic (Table to Class)	R2O language	R2O mapping document		SPARQL→SQL→ RDB	Generic	Enables	Potentially Multiple
4. Tirmizi et. al 2008	Automatic (Table to Class, SQL-DDL to RDF)	First Order Logic	None specified	Static	None Specified	Generic	No	None
5. SOAM (Li et al., 2005)	Automatic (Table to Class) with user input	Logic Rules	Implemented as part of system	Static	Potentially SPARQL (on generated populated ontology)	Generic (Case Study: Economics)	No	None
6. Sahoo et al., 2008	Manual/Semi-Automatic (Domain Semantics-driven)	XPath expressions	XSLT document	Static	SPARQL	Life Sciences	Yes	Test included five (Gene, Biological Pathway)
7. Byrne, 2008	Manual/Semi-Automatic (Domain Semantics-driven)	SKOS vocabulary	RDF document	Static	SPARQL	Cultural Heritage	No	None
8. Green et. al, 2008	Manual/Semi-Automatic (Domain Semantics-driven)	D2RQ language	D2RQ mapping file	Dynamic	SPARQL→SQL→ RDB	Ordnance Survey	Yes	Mutltiple
9. Virtuoso RDF View (Blakeley, 2007)	Both (user-specified)	SPASQL-based Meta Schema Language	Quad Storage	Both	Both	Generic	Enables	Potentially Multiple
10. D2RQ (Bizer et al., 2007)	Both (user-specified)	D2RQ language	D2RQ mapping file	Both	Both	Generic	Enables	Potentially Multiple
11. R2O (Barrasa et al., 2006)	Both (user-specified)	R2O language	R2O mapping document	Both	Both	Generic	Enables	Potentially Multiple
12. Dartgrid (Wu et al., 2006)	Automatic (Table to Class)	XML File	Visualized Mapping tool	Dynamic	SPARQL→SQL→ RDB (Provide search and query interface)	Life Science (Traditional Chinese Medicine, TCM)	Yes	Test included databases for herb, compound formulas, disease, drug, TCM treatment.
13. RDBtoOnto (Cerbah, 2008)	Automatic (Table to Class, allows user intervention)	Constraint rules	Not explicitly stored	Static	Potentially SPARQL (on generated populated ontology)	Generic	No	None
14. Asio Tools	Automatic (Table to	OWL Full based language	File based	Both	SPARQL→SQL→ RDB	Generic	Enables	Potentially Multiple

Fig. 11: Comparison of approaches from [134].

primary intended use of R2RML but may be desirable for importing linked data into relational stores. This is possible if the constituent mappings and underlying SQL objects constitute updateable views in the SQL sense.

Data integration is often mentioned as a motivating use case for the adoption of RDF. This integration will very often be between relational databases which have logical entities in common, each with its local schema and identifiers.Thus, we expect to see relational to RDF mapping use cases involving the possibility of a triple coming from multiple sources. This does not present any problem if RDF is being extracted but does lead to complications if SPARQL queries are mapped into SQL. In specific, one will end up with potentially very long queries consisting of joins of unions. Most of the joins between terms of the unions will often be provably empty and can thus be optimized away. This capability however requires the mapping language to be able to express metadata about mappings, i.e. that IRIs coming from one place are always disjoint from IRIs coming from another place. Without such metadata optimizing SPARQL to SQL translation is not possible, which will significantly limit the possibility of querying collections of SQL databases through a SPARQL end point without ETL-ing the mapped RDF into an RDF store.

RDF is emerging as a format for interoperable data publishing. This does not entail that RDF were preferable as a data warehousing model. Besides, for large warehouses, RDF is not cost competitive with relational technology, even though projects such as LOD2 and LDBC expect to narrow this gap (see, e.g., [101,102] for recent SPARQL benchmarks). Thus it follows that on the fly mapping of SPARQL to SQL will be important. Regardless of the relative cost or performance of relational or RDF technology, it is not a feasible proposition to convert relational warehouses to RDF in general, rather existing investments must be protected and reused. Due to these reasons, R2RML will have to evolve in the direction of facilitating querying of federated relational resources.

Supervised Extraction Example: Sparqlify The challenges encountered with large scale relational data sources LinkedGeoData [12,144] indicate that ETL style approaches based on the conversion of all underlying data to RDF have severe deficiencies. For instance, the RDF conversion process is very time consuming for large-scale, crowdsourced data. Furthermore, changes in data modelling require many changes in the extracted RDF data or the creation of a completely new dump. In summary, the ETL approach is not sufficiently flexible for very large and frequently changing data. It seems preferable to establish virtual RDF views over the existing relational database. In contrast to other tools, such as *D2R* and *Virtuoso RDF views*, Sparqlify converts each SPARQL query to a single SQL query. This allows all optimisations of the underlying database to be applied and can lead to better scalability.

Figure 12 shows the query rewriting workflow in Sparqlify. The rationale of Sparqlify is to leave the schema of the underlying relational database schema unmodified and define RDF views over it. SPARQL queries can then be written against those views, which are expressed in the *Sparqlify-ML* (mapping language). Sparqlify-ML is easy to learn for users, who are experienced in SPARQL and SQL and more compact than other syntactic variants such as R2RML. The left part of Figure 12 shows all steps, which are performed to answer a query. First, the query is converted into an algebra expression.

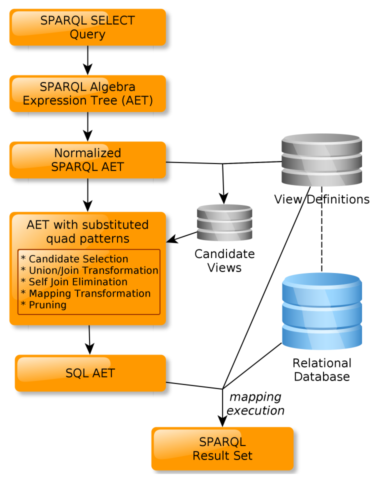


Fig. 12: The Sparqlify concepts and query rewriting workflow.

This expression is subsequently converted to a normal form. Given the query patterns, relevant Sparqlify-ML views need to be detected. After this is done, the algebra expression is rewritten to include those relevant views. In a next step, optimisations on the algebra expression are performed to improve efficiency. Finally, this algebra expression can be transformed to an SQL algebra expression. For accomplishing this, we define a general relational algebra for RDB-to-RDF mappings. The SQL query, which was obtained, is executed against the relational database. Using the defined mappings, the SQL result set returned by the relational database can be converted to a SPARQL result set.

All of the above steps are explained in detail throughout the next sections.The main contribution of the Sparqlify project is a formalization, which goes beyond previous work by being capable to push the complete query execution using a single SQL query into the DBMS.

4 Authoring with Semantic Wikis

Semantic Wikis are an extension to conventional, text-based Wikis. While in conventional Wikis pages are stored as blocks of text using a special Wiki markup for structuring the display of the text and adding links to other pages, semantic Wikis aim at adding rich structure to the information itself. To this end, two initially orthogonal approaches have been used: a) extending the markup language to allow semantic annotations and links with meaning or b) building the Wiki software directly with structured information in mind. Nowadays, both approaches have somewhat converged, for instance Semantic MediaWiki [76] also provides forms for entering structured data (see Figure 13). Characteristics of both approaches are summarized in Table 2 for the two prototypical representatives of both approaches, i.e. Semantic MediaWiki and OntoWiki.

	Semantic MediaWiki	**OntoWiki**
Managed entities	Articles	Resources
Editing	Wiki markup	Forms
Atomic element	Text blob	Statement

Table 2: Conceptual differences between Semantic MediaWiki and OntoWiki.

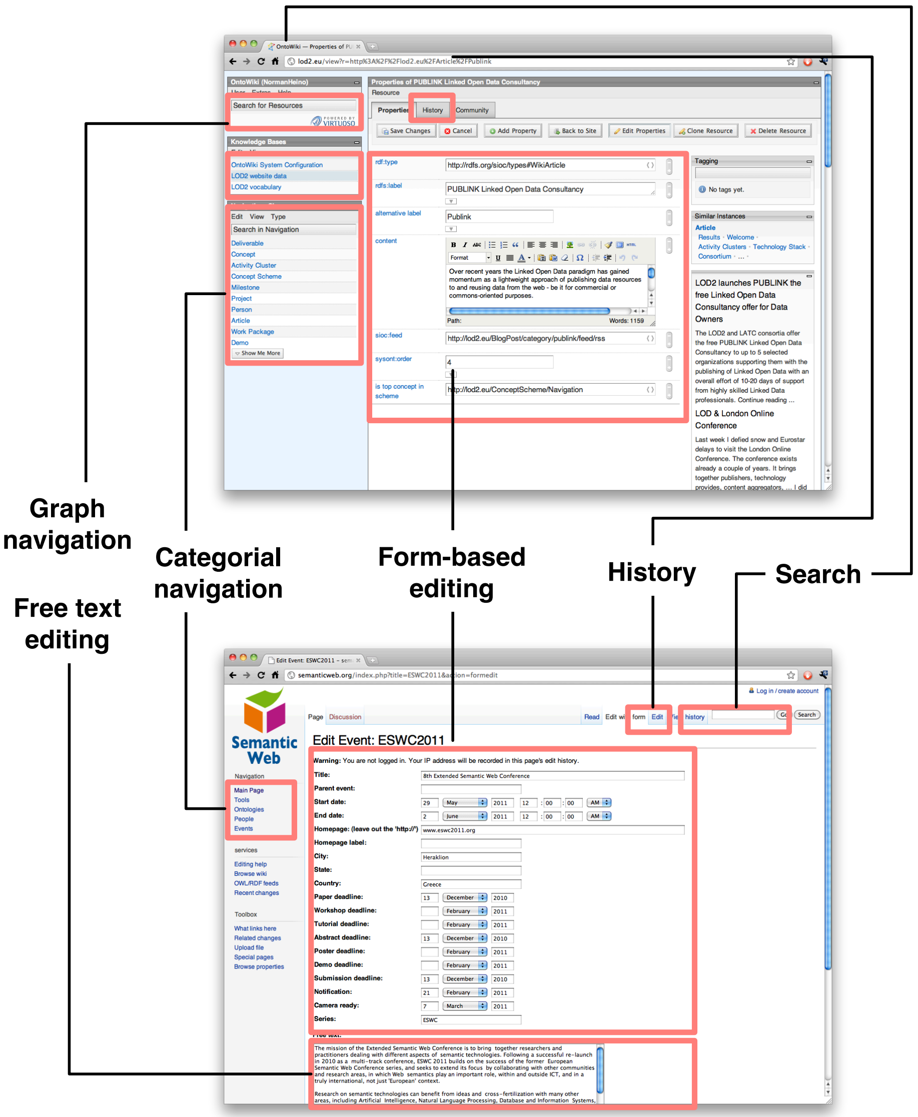
Extending Wikis with Semantic Markup. The benefit of a Wiki system comes from the amount of interlinking between Wiki pages. Those links clearly state a relationship between the linked-to and the linking page. However, in conventional Wiki systems this relationship cannot be made explicit. Semantic Wiki systems therefore add a means to specify typed relations by extending the Wiki markup with semantic (i.e. typed) links. Once in place, those links form a knowledge base underlying the Wiki which can be used to improve search, browsing or automatically generated lists and category pages. Examples of approaches for extending Wikis with semantic markup can be found in [76,137,14,121,142]. They represent a straightforward combination of existing Wiki systems and the Semantic Web knowledge representation paradigms. Yet, we see the following obstacles:

***Usability*:** The main advantage of Wiki systems is their unbeatable usability. Adding more and more syntactic possibilities counteracts ease of use for editors.

***Redundancy*:** To allow the answering of real-time queries to the knowledge base, statements have to be additionally kept in a triple store. This introduces a redundancy, which complicates the implementation.

***Evolution*:** As a result of storing information in both Wiki texts and triple store, supporting evolution of knowledge is difficult.

Wikis for Editing Structured Data. In contrast to text-based systems, Wikis for structured data – also called Data Wikis – are built on a structured model of the data being



Properties of PUBLINK Linked Open Data Consultancy

rdf:type: http://rdfs.org/sioc/types#WikiArticle
rdfs:label: PUBLINK Linked Open Data Consultancy
alternative label: Publink
content: Over recent years the Linked Open Data paradigm has gained momentum as a lightweight approach of publishing data resources to and reusing data from the web - be it for commercial or commons-oriented purposes.
sioc:feed: http://lod2.eu/BlogPost/category/publink/feed/rss
sysont:order: 4
is top concept in scheme: http://lod2.eu/ConceptScheme/Navigation

Edit Event: ESWC2011

Warning: You are not logged in. Your IP address will be recorded in this page's edit history.

Title:	8th Extended Semantic Web Conference
Parent event:	
Start date:	29 May 2011 12 : 00 : 00 AM
End date:	2 June 2011 12 : 00 : 00 AM
Homepage: (leave out the 'http://')	www.eswc2011.org
Homepage label:	
City:	Heraklion
State:	
Country:	Greece
Paper deadline:	13 December 2010
Workshop deadline:	February 2011
Tutorial deadline:	February 2011
Abstract deadline:	13 December 2010
Poster deadline:	February 2011
Demo deadline:	February 2011
Submission deadline:	13 December 2010
Notification:	21 February 2011
Camera ready:	7 March 2011
Series:	ESWC

The mission of the Extended Semantic Web Conference is to bring together researchers and practitioners dealing with different aspects of semantic technologies. Following a successful re-launch in 2010 as a multi-track conference, ESWC 2011 builds on the success of the former European Semantic Web Conference series, and seeks to extend its focus by collaborating with other communities and research areas, in which Web semantics play an important role, within and outside ICT, and in a truly international, not just 'European' context.

Research on semantic technologies can benefit from ideas and cross-fertilization with many other areas, including Artificial Intelligence, Natural Language Processing, Database and Information Systems,

Fig. 13: Comparison of Semantic MediaWiki and OntoWiki GUI building blocks.

edited. The Wiki software can be used to add instances according to the schema or (in some systems) edit the schema itself. One of those systems is OntoWiki[7] [9] which bases its data model on RDF. This way, both schema and instance data are represented using the same low-level model (i.e. statements) and can therefore be handled identically by the Wiki.

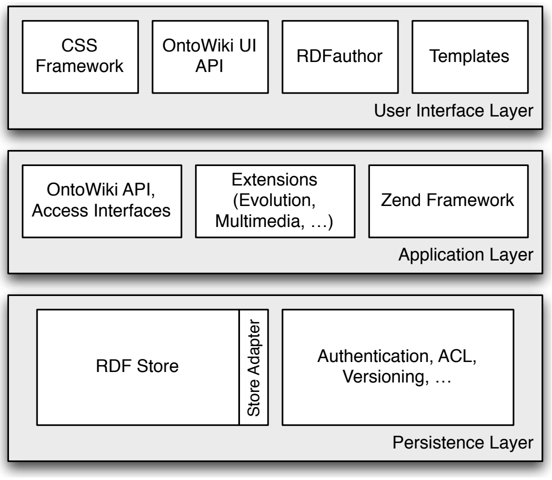


Fig. 14: Overview of OntoWiki's architecture with extension API and Zend web framework (modified according to [54]).

4.1 OntoWiki - a Semantic Data Wiki

OntoWiki started as an RDF-based data wiki with emphasis on collaboration but has meanwhile evolved into a comprehensive framework for developing Semantic Web applications [54]. This involved not only the development of a sophisticated extension interface allowing for a wide range of customizations but also the addition of several access and consumption interfaces allowing OntoWiki installations to play both a provider and a consumer role in the emerging Web of Data.

OntoWiki is inspired by classical Wiki systems, its design, however, (as mentioned above) is independent and complementary to conventional Wiki technologies. In contrast to other semantic Wiki approaches, in OntoWiki text editing and knowledge engineering (i. e. working with structured knowledge bases) are not mixed. Instead, OntoWiki directly applies the Wiki paradigm of "making it easy to correct mistakes, rather than making it hard to make them" [89] to collaborative management of structured knowledge. This paradigm is achieved by interpreting knowledge bases as *information maps* where every node is represented visually and interlinked to related resources.

[7] Available at: `http://ontowiki.net`

Furthermore, it is possible to enhance the knowledge schema gradually as well as the related instance data agreeing on it. As a result, the following requirements and corresponding features characterize OntoWiki:

Intuitive display and editing of instance data should be provided in generic ways, yet enabling domain-specific presentation of knowledge.
Semantic views allow the generation of different views and aggregations of the knowledge base.
Versioning and evolution provides the opportunity to track, review and roll-back changes selectively.
Semantic search facilitates easy-to-use full-text searches on all literal data, search results can be filtered and sorted (using semantic relations).
Community support enables discussions about small information chunks. Users are encouraged to vote about distinct facts or prospective changes.
Online statistics interactively measures the popularity of content and activity of users.
Semantic syndication supports the distribution of information and their integration into desktop applications.

OntoWiki enables the easy creation of highly structured content by distributed communities. The following points summarize some limitations and weaknesses of OntoWiki and thus characterize the application domain:

***Environment*:** OntoWiki is a Web application and presumes all collaborators to work in a Web environment, possibly distributed.
***Usage Scenario*:** OntoWiki focuses on knowledge engineering projects where a single, precise usage scenario is either initially (yet) unknown or not (easily) definable.
***Reasoning*:** Application of reasoning services was (initially) not the primary focus.

4.2 Generic and Domain-specific Views

OntoWiki can be used as a tool for presenting, authoring and managing knowledge bases adhering to the RDF data model. As such, it provides generic methods and views, independent of the domain concerned. Two generic views included in OntoWiki are the resource view and the list view. While the former is generally used for displaying all known information about a resource, the latter can present a set of resources, typically instances of a certain concept. That concept must not necessarily be explicitly defined as `rdfs:Class` or `owl:Class` in the knowledge base. Via its faceted browsing, OntoWiki allows the construction of complex concept definitions, with a pre-defined class as a starting point by means of property value restrictions. These two views are sufficient for browsing and editing all information contained in a knowledge base in a generic way. For domain-specific use cases, OntoWiki provides an easy-to-use extension interface that enables the integration of custom components. By providing such a custom view, it is even possible to hide completely the fact that an RDF knowledge base is worked on. This permits OntoWiki to be used as a data-entry frontend for users with a less profound knowledge of Semantic Web technologies.

4.3 Workflow

With the use of RDFS [27] and OWL [125] as ontology languages, resource definition is divisible into different layers: a terminology box for conceptual information (i. e. classes and properties) and an assertion box for entities using the concepts defined (i. e. instances). There are characteristics of RDF which, for end users, are not easy to comprehend (e. g. *classes* can be defined as *instances* of `owl:Class`). OntoWiki's user interface, therefore, provides elements for these two layers, simultaneously increasing usability and improving a user's comprehension for the structure of the data. After starting and logging in into OntoWiki with registered user credentials, it is possible to select one of the existing ontologies. The user is then presented with general information about the ontology (i. e. all statements expressed about the knowledge base as a resource) and a list of defined classes, as part of the conceptual layer.

After starting and logging in into OntoWiki with registered user credentials, it is possible to select one of the existing knowledge bases. The user is then presented with general information about the ontology (i. e. all statements expressed about the knowledge base as a resource) and a list of defined classes, as part of the conceptual layer. By selecting one of these classes, the user obtains a list of the class' instances. OntoWiki applies basic `rdfs:subClassOf` reasoning automatically. After selecting an instance from the list – or alternatively creating a new one – it is possible to manage (i. e. insert, edit and update) information in the details view.OntoWiki focuses primarily on the assertion layer, but also provides ways to manage resources on the conceptual layer. By enabling the visualization of schema elements, called *System Classes* in the OntoWiki nomenclature, conceptional resources can be managed in a similar fashion as instance data.

4.4 Authoring

Semantic content in OntoWiki is represented as resource descriptions. Following the RDF data model representing one of the foundations of the Semantic Web vision, resource descriptions are represented (at the lowest level) in the form of *statements*. Each of these statements (or triples) consist of a *subject* which identifies a resource as well as a *predicate* and an *object* which together represent data about said resource in a fashion reminiscent of key-value pairs. By means of RDFa [2], these statements are retained in the HTML view (i.e. user interface) part and are thus accessible to client-side techniques like JavaScript.

Authoring of such content is based on said client-side representation by employing the RDFauthor approach [147]: views are declared in terms of the model language (RDF) which allows the underlying model be restored. Based on this model, a user interface can be generated with the model being providing all the domain knowledge required to do so. The RDFauthor system provides an extensible set of authoring widgets specialized for certain editing tasks. RDFauthor was also extended by adding capabilities for automatically translating literal object values between different languages. Since the semantic context is known to the system, these translation functionality can be bound to arbitrary characteristics of the data (e. g. to a certain property or a missing language).

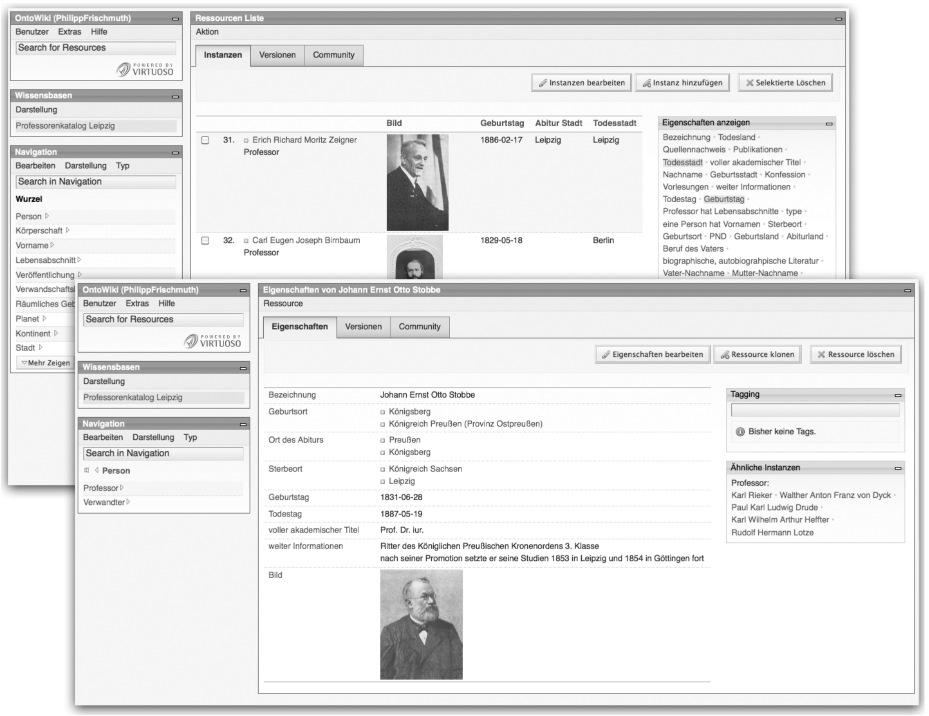

OntoWiki (PhilippFrischmuth)
Benutzer Extras Hilfe
Search for Resources
Wissensbasen
Darstellung
Professorenkatalog Leipzig
Navigation
Bearbeiten Darstellung Typ
Search in Navigation
Wurzel
Person
Körperschaft
Vorname
Lebensabschnitt
Veröffentlichung
Verwandschafts
Räumliches Geb
Planet
Kontinent
Stadt
Mehr Zeigen
Ressourcen Liste
Aktion
Instanzen
Versionen
Community
Instanzen bearbeiten
Instanz hinzufügen
Selektierte Löschen
Eigenschaften anzeigen
Bezeichnung · Todesland · Quellennachweis · Publikationen · Todesstadt · voller akademischer Titel · Nachname · Geburtsstadt · Konfession · Vorlesungen · weiter Informationen · Todestag · Geburtstag · Professor hat Lebensabschnitte · type · eine Person hat Vornamen · Sterbeort · Geburtsort · PND · Geburtsland · Abiturland · Beruf des Vaters · biographische, autobiograhpische Literatur · Vater-Nachname · Mutter-Nachname ·
Eigenschaften von Johann Ernst Otto Stobbe
Ressource
Eigenschaften
Versionen
Community
Eigenschaften bearbeiten
Ressource klonen
Ressource löschen
Person
Professor
Verwandter
Tagging
Bisher keine Tags.
Ähnliche Instanzen
Professor:
Karl Rieker · Walther Anton Franz von Dyck · Paul Karl Ludwig Drude · Karl Wilhelm Arthur Heffter · Rudolf Hermann Lotze

		Bild	Geburtstag	Abitur Stadt	Todesstadt
31.	Erich Richard Moritz Zeigner Professor		1886-02-17	Leipzig	Leipzig
32.	Carl Eugen Joseph Birnbaum Professor		1829-05-18		Berlin

Bezeichnung	Johann Ernst Otto Stobbe
Geburtsort	Königsberg Königreich Preußen (Provinz Ostpreußen)
Ort des Abiturs	Preußen Königsberg
Sterbeort	Königreich Sachsen Leipzig
Geburtstag	1831-06-28
Todestag	1887-05-19
voller akademischer Titel	Prof. Dr. iur.
weiter Informationen	Ritter des Königlichen Preußischen Kronenordens 3. Klasse nach seiner Promotion setzte er seine Studien 1853 in Leipzig und 1854 in Göttingen fort
Bild	

Fig. 15: OntoWiki views: (background) A tabular list view, which contains a filtered list of resources highlighting some specific properties of those resources and (foreground) a resource view which allows to tag and comment a specific resource as well as editing all property values.

Versioning & Evolution. As outlined in the wiki principles, keeping track of all changes is an important task in order to encourage user participation. OntoWiki applies this concept to RDF-based knowledge engineering in that all changes are tracked on the statement level [10]. These low-level changes can be grouped to reflect application- and domain-specific tasks involving modifications to several statements as a single versioned item. Provenance information as well as other metadata (such as time, user or context) of a particular changeset can be attached to each individual changeset. All changes on the knowledge base can be easily reviewed and rolled-back if needed. The loosely typed data model of RDF encourages continuous evolution and refinement of knowledge bases. With *EvoPat*, OntoWiki supports this in a declarative, pattern-based manner (cf. **??**).

4.5 Access Interfaces

In addition to human-targeted graphical user interfaces, OntoWiki supports a number of machine-accessible data interfaces. These are based on established Semantic Web standards like SPARQL or accepted best practices like publication and consumption of Linked Data.

SPARQL Endpoint. The SPARQL recommendation not only defines a query language for RDF but also a protocol for sending queries to and receiving results from remote endpoints[8]. OntoWiki implements this specification, allowing all resources managed in an OntoWiki be queried over the Web. In fact, the aforementioned RDFauthor authoring interface makes use of SPARQL to query for additional schema-related information, treating OntoWiki as a remote endpoint in that case.

Linked Data. Each OntoWiki installation can be part of the emerging Linked Data Web. According to the Linked Data publication principles (cf. section 2), OntoWiki makes all resources accessible by its IRI (provided, the resource's IRI is in the same namespace as the OntoWiki instance). Furthermore, for each resource used in OntoWiki additional triples can be fetches if the resource is de-referenceable.

Semantic Pingback. Pingback is an established notification system that gained wide popularity in the blogsphere. With Semantic Pingback [146], OntoWiki adapts this idea to Linked Data providing a *notification mechanism* for resource usage. If a Pingback-enabled resource is mentioned (i. e. linked to) by another party, its pingback server is notified of the usage. Provided, the Semantic Pingback extension is enabled all resources used in OntoWiki are pinged automatically and all resources defined in OntoWiki are Pingback-enabled.

4.6 Exploration Interfaces

For exploring semantic content, OntoWiki provides several exploration interfaces that range from generic views over search interfaces to sophisticated querying capabilities for more RDF-knowledgable users. The subsequent paragraphs give an overview of each of them.

Knowledge base as an information map. The compromise between, on the one hand, providing a generic user interface for arbitrary RDF knowledge bases and, on the other hand, aiming at being as intuitive as possible is tackled by regarding knowledge bases as *information maps*. Each node at the information map, i. e. RDF resource, is represented as a Web accessible page and interlinked to related digital resources. These Web pages representing nodes in the information map are divided into three parts: a left sidebar, a main content section and a right sidebar. The left sidebar offers the selection of content to display in the main content section. Selection opportunities include the set of available knowledge bases, a hierarchical browser and a full-text search.

Full-text search. The full-text search makes use of special indexes (mapped to proprietary extensions to the SPARQL syntax) if the underlying knowledge store provides this feature, else, plain SPARQL string matching is used. In both cases, the resulting

[8] `http://www.w3.org/TR/rdf-sparql-protocol/`

SPARQL query is stored as an object which can later be modified (e. g. have its filter clauses refined). Thus, full-text search is seamlessly integrated with faceted browsing (see below).

Content specific browsing interfaces. For domain-specific use cases, OntoWiki provides an easy-to-use extension interface that enables the integration of custom components. By providing such a custom view, it is even possible to hide completely the fact that an RDF knowledge base is worked on. This permits OntoWiki to be used as a data-entry frontend for users with a less profound knowledge of Semantic Web technologies.

Faceted-browsing. Via its faceted browsing, OntoWiki allows the construction of complex concept definitions, with a pre-defined class as a starting point by means of property value restrictions. These two views are sufficient for browsing and editing all information contained in a knowledge base in a generic way.

Query-builder. OntoWiki serves as a SPARQL endpoint, however, it quickly turned out that formulating SPARQL queries is too tedious for end users. In order to simplify the creation of queries, we developed the *Visual Query Builder*[9] (VQB) as an OntoWiki extension, which is implemented in JavaScript and communicates with the triple store using the SPARQL language and protocol. VQB allows to visually create queries to the stored knowledge base and supports domain experts with an intuitive visual representation of query and data. Developed queries can be stored and added via drag-and-drop to the current query. This enables the reuse of existing queries as building blocks for more complex ones.

4.7 Applications

Catalogous Professorum. The World Wide Web, as an ubiquitous medium for publication and exchange, already significantly influenced the way historians work: the online availability of catalogs and bibliographies allows to efficiently search for content relevant for a certain investigation; the increasing digitization of works from historical archives and libraries, in addition, enables historians to directly access historical sources remotely. The capabilities of the Web as a medium for collaboration, however, are only starting to be explored. Many, historical questions can only be answered by combining information from different sources, from different researchers and organizations. Also, after original sources are analyzed, the derived information is often much richer, than can be captured by simple keyword indexing. These factors pave the way for the successful application of knowledge engineering techniques in historical research communities.

In [130] we report about the application of an adaptive, semantics-based knowledge engineering approach using OntoWiki for the development of a prosopographical knowledge base. In prosopographical research, historians analyze common characteristics of historical groups by studying statistically relevant quantities of individual biographies. Untraceable periods of biographies can be determined on the basis of such

[9] `http://aksw.org/Projects/OntoWiki/Extension/VQB`

accomplished analyses in combination with statistically examinations as well as patterns of relationships between individuals and their activities.

In our case, researchers from the historical seminar at Universität Leipzig aimed at creating a prosopographical knowledge base about the life and work of professors in the 600 years history of Universität Leipzig ranging from the year 1409 till 2009 - the *Catalogus Professorum Lipsiensis* (CPL). In order to enable historians to collect, structure and publish this prosopographical knowledge an ontological knowledge model was developed and incrementally refined over a period of three years. The community of historians working on the project was enabled to add information to the knowledge base using an adapted version of OntoWiki. For the general public, a simplified user interface[10] is dynamically generated based on the content of the knowledge base. For access and exploration of the knowledge base by other historians a number of access interfaces was developed and deployed, such as a graphical SPARQL query builder, a relationship finder and plain RDF and Linked Data interfaces. As a result, a group of 10 historians supported by a much larger group of volunteers and external contributors collected information about 1,300 professors, 10,000 associated periods of life, 400 institutions and many more related entities.

The benefits of the developed knowledge engineering platform for historians are twofold: Firstly, the collaboration between the participating historians has significantly improved: The ontological structuring helped to quickly establish a common understanding of the domain. Collaborators within the project, peers in the historic community as well as the general public were enabled to directly observe the progress, thus facilitating peer-review, feedback and giving direct benefits to the contributors. Secondly, the ontological representation of the knowledge facilitated original historical investigations, such as historical social network analysis, professor appointment analysis (e.g. with regard to the influence of cousin-hood or political influence) or the relation between religion and university. The use of the developed model and knowledge engineering techniques is easily transferable to other prosopographical research projects and with adaptations to the ontology model to other historical research in general. In the long term, the use of collaborative knowledge engineering in historian research communities can facilitate the transition from largely individual-driven research (where one historian investigates a certain research question solitarily) to more community-oriented research (where many participants contribute pieces of information in order to enlighten a larger research question). Also, this will improve the reusability of the results of historic research, since knowledge represented in structured ways can be used for previously not anticipated research questions.

OntoWiki Mobile. As comparatively powerful mobile computing devices are becoming more common, mobile web applications have started gaining in popularity. An important feature of these applications is their ability to provide *offline functionality* with local updates for later synchronization with a web server. The key problem here is the reconciliation, i. e. the problem of potentially *conflicting updates* from *disconnected clients*. Another problem current mobile application developers face is the plethora of mobile application development platforms as well as the incompatibilities between

[10] Available at: `http://www.uni-leipzig.de/unigeschichte/professorenkatalog/`

them. *Android* (Google), *iOS* (Apple), *Blackberry OS* (RIM), *WebOS* (HP/Palm), *Symbian* (Nokia) are popular and currently widely deployed platforms, with many more proprietary ones being available as well. As a consequence of this fragmentation, realizing a special purpose application, which works with many or all of these platforms is extremely time consuming and inefficient due to the large amount of duplicate work required.

The W3C addressed this problem, by enriching HTML in its 5th revision with access interfaces to local storage (beyond simple cookies) as well as a number of devices and sensors commonly found on mobile devices (e. g. GPS, camera, compass etc.). We argue, that in combination with semantic technologies these features can be used to realize a *general purpose*, mobile collaboration platform, which can support the long tail of mobile special interest applications, for which the development of individual tools would not be (economically) feasible.

In [38] we present the *OntoWiki Mobile* approach realizing a mobile semantic collaboration platform based on the OntoWiki. It comprises specifically adopted user interfaces for browsing, faceted navigation as well as authoring of knowledge bases. It allows users to collect instance data and refine the structured knowledge bases on-the-go. OntoWiki Mobile is implemented as an *HTML5 web application*, thus being completely mobile device platform independent. In order to allow offline use in cases with restricted network coverage (or in order to avoid roaming charges) it uses the novel HTML5 local storage feature for replicating parts of the knowledge base on the mobile device. Hence, a crucial part of OntoWiki Mobile is the advanced conflict resolution for RDF stores. The approach is based on a combination of the EvoPat [131] method for data evolution and ontology refactoring along with a versioning system inspired by distributed version control systems like Git. OntoWiki Mobile is a generic, application domain agnostic tool, which can be utilized in a wide range of very different usage scenarios ranging from instance acquisition to browsing of semantic data on the go. Typical OntoWiki Mobile usage scenarios are settings where users need to author and access semantically structured information on the go or in settings where users are away from regular power supply and restricted to light-weight equipment (e. g. scientific expeditions).

Semantics-based Requirements Engineering. Semantic interoperability, linked data, and a shared conceptual foundation become increasingly important prerequisites in software development projects that are characterized by spatial dispersion, large numbers of stakeholders, and heterogeneous development tools. The SoftWiki OntoWiki extension [92] focuses specifically on semantic collaboration with respect to requirements engineering. Potentially very large and spatially distributed groups of stakeholders, including developers, experts, managers, and average users, shall be enabled to collect, semantically enrich, classify, and aggregate software requirements. OntoWiki is used to support collaboration as well as interlinking and exchange of requirements data. To ensure a shared conceptual foundation and semantic interoperability, we developed the SoftWiki Ontology for Requirements Engineering (SWORE) that defines core concepts of requirement engineering and the way they are interrelated. For instance, the ontology defines frequent relation types to describe requirements interdependencies such as details, conflicts, related to, depends on, etc. The flexible SWORE design allows for easy extension. Moreover, the requirements can be linked to external resources, such as publicly available domain knowledge or company-specific policies. The whole process is called semantification of requirements. It is envisioned as an evolutionary process: The requirements are successively linked to each other and to further concepts in a collaborative way, jointly by all stakeholders. Whenever a requirement is formulated, reformulated, analyzed, or exchanged, it might be semantically enriched by the respective participant.

5 Linking

The fourth Linked Data Principle, i.e., "Include links to other URIs, so that they can discover more things" (cf. section 2) is the most important Linked Data principle as it enables the paradigm change from data silos to interoperable data distributed across the Web. Furthermore, it plays a key role in important tasks such as cross-ontology question answering [22,93], large-scale inferences [150,98] and data integration [94,19]. Yet, while the number of triples in Linked Data sources increases steadily and has surpassed 31 billions[11], links between knowledge bases still constitute less than 5% of these triples. The goal of linking is to tackle this sparseness so as to transform the Web into a platform for data and information integration as well as for search and querying.

5.1 Link Discovery

Linking can be generally defined as *connecting things that are somehow related*. In the context of Linked Data, the idea of linking is especially concerned with establishing typed links between entities (i.e., classes, properties or instances) contained in knowledge bases. Over the last years, several frameworks have been developed to address the lack of typed links between the different knowledge bases on the Linked Data web. Overall, two main categories of frameworks that aim to achieve this goal can be differentiated. The first category implements *ontology matching* techniques and aims to establish links between the ontologies underlying two data sources. The second and more prominent category of approaches, dubbed *instance matching* approaches (also called linking or link discovery approaches), aims to discover links between instances contained in two data sources. It is important to notice that while ontology and instance matching are similar to schema matching [127,126] and record linkage [161,37,25] respectively (as known in the research area of databases), linking on the Web of Data is a more generic and thus more complex task, as it is not limited to finding equivalent entities in two knowledge bases. Rather, it aims at finding semantically related entities and establishing typed links between them, most of these links being imbued with formal properties (e.g., transitivity, symmetry, etc.) that can be used by reasoners and other application to infer novel knowledge. In this section, we will focus on the discovery of links between instances and use the term link discovery as name for this process. An overview of ontology matching techniques is given in [41].

Formally, link discovery can be defined as follows:

Definition 3 (Link Discovery). *Given two sets S (source) and T (target) of instances, a (complex) semantic similarity measure $\sigma : S \times T \to [0,1]$ and a threshold $\theta \in [0,1]$, the goal of link discovery task is to compute the set $M = \{(s,t), \sigma(s,t) \geq \theta\}$.*

In general, the similarity function used to carry out a link discovery task is described by using a *link specification* (sometimes called *linkage decision rule* [67]).

[11] `http://lod-cloud.net/`

5.2 Challenges

Two key challenges arise when trying to discover links between two sets of instances: the computational complexity of the matching task *per se* and the selection of an appropriate link specification. The first challenge is intrinsically related to the link discovery process. The time complexity of a matching task can be measured by the number of comparisons necessary to complete this task. When comparing a source knowledge base S with a target knowledge base T, the completion of a matching task requires a-priori $O(|S||T|)$ comparisons, an impractical proposition as soon as the source and target knowledge bases become large. For example, discovering duplicate cities in DBpedia [6] alone would necessitate approximately 0.15×10^9 similarity computations. Hence, the provision of time-efficient approaches for the reduction of the time complexity of link discovery is a key requirement to instance linking frameworks for Linked Data.

The second challenge of the link discovery process lies in the selection of an appropriate link specification. The configuration of link discovery frameworks is usually carried out manually, in most cases simply by guessing. Yet, the choice of a suitable link specification measure is central for the discovery of satisfactory links. The large number of properties of instances and the large spectrum of measures available in literature underline the complexity of choosing the right specification manually[12]. Supporting the user during the process of finding the appropriate similarity measure and the right properties for each mapping task is a problem that still needs to be addressed by the Linked Data community. Methods such as supervised and active learning can be used to guide the user in need of mapping to a suitable linking configuration for his matching task. In the following, we give a short overview of existing frameworks for Link Discovery on the Web of Data. Subsequently, we present a time-efficient framework for link discovery in more detail and show how it can detect link specifications using active learning.

5.3 Approaches to Link Discovery

Current frameworks for link discovery can be subdivided into two main categories: *domain-specific* and *universal* frameworks. Domain-specific link discovery frameworks aim at discovering links between knowledge bases from a particular domain. One of the first domain-specific approaches to carry out instance linking for Linked Data was implemented in the *RKBExplorer*[13] [49] with the aim of discovering links between entities from the domain of academics. Due to the lack of data available as Linked Data, the RKBExplorer had to extract RDF from heterogeneous data source so as to populate its knowledge bases with instances according to the AKT ontology[14]. Especially, instances of persons, publications and institutions were retrieved from several major metadata websites such as ACM and DBLP. The linking was implemented by the so-called Consistent Reference Service (CRS) which linked equivalent entities by comparing properties including their type and label. So far, the CRS is limited to linking

[12] The SimMetrics project (`http://simmetrics.sf.net`) provides an overview of strings similarity measures.

[13] `http://www.rkbexplorer.com`

[14] `http://www.aktors.org/publications/ontology/`

objects in the knowledge bases underlying the RKBExplorer and cannot be used for other tasks without further implementation.

Another domain-specific tool is GNAT [128], which was developed for the music domain. It implements several instance matching algorithms of which the most sophisticated, the online graph matching algorithm (OGMA), applies a similarity propagation approach to discover equivalent resources. The basic approach implemented by OGMA starts with a single resource $s \in S$. Then, it retrieves candidate matching resources $t \in T$ by comparing properties such as `foaf:name` for artists and `dc:title` for albums. If $\sigma(s,t) \geq \theta$, then the algorithm terminates. In case a disambiguation is needed, the resourced related to s and t in their respective knowledge bases are compared and their similarity value is cumulated to recompute $\sigma(s,t)$. This process is iterated until a mapping resource for s is found in T or no resource matches.

Universal link discovery frameworks are designed to carry out mapping tasks independently from the domain of the source and target knowledge bases. For example, RDF-AI [138], a framework for the integration of RDF data sets, implements a five-step approach that comprises the preprocessing, matching, fusion, interlinking and post-processing of data sets. RDF-AI contains a series of modules that allow for computing instances matches by comparing their properties. Especially, it contains translation modules that allow to process the information contained in data sources before mapping. By these means, it can boost the precision of the mapping process. These modules can be configured by means of XML-files. RDF-AI does not comprise means for querying distributed data sets via SPARQL[15]. In addition, it suffers from not being time-optimized. Thus, mapping by using this tool can be very time-consuming.

A time-optimized approach to link discovery is implemented by the LIMES framework [111,110,115] (Link Discovery Framework for metric spaces) .[16] The idea behind the LIMES framework is to use the mathematical characteristics of similarity and distance measures to reduce the number of computations that have to be carried out by the system without losing any link. For example, LIMES can make use of the fact that the edit distance is a distance metric to approximate distances without having to compute them [111]. Moreover, it implements the reductio-ratio-optimal space tiling algorithm $\mathcal{HR}^3$ to compute similarities in affine spaces with Minkowski measures [109]. In contrast to other frameworks (of which most rely on blocking), LIMES relies on time-efficient set operators to combine the results of these algorithms efficiently and has been shown to outperform the state of the art by these means [110]. Moreover, LIMES implements unsupervised and supervised machine learning approaches for detecting high-quality link specifications [115,116].

Another link discovery framework is SILK [155]. SILK implements several approaches to minimize the time necessary for mapping instances from knowledge bases. In addition to implementing rough index pre-matching to reach a quasi-linear time-complexity, SILK also implements a lossless blocking algorithm called MultiBlock [68] to reduce its overall runtime. The approach relies on generating overlapping blocks of instances and only comparing pairs of instances that are located in the same block.

[15] `http://www.w3.org/TR/rdf-sparql-query/`

[16] `http://limes.sf.net`. A graphical user interface can be found at `http://saim.aksw.org`.

Moreover, SILK provides supervised machine learning approaches for link discovery [69].

It is important to notice that the task of discovering links between knowledge bases is related with record linkage [161,37] and de-duplication [25]. The database community has produced a vast amount of literature on efficient algorithms for solving these problems. Different blocking techniques such as standard blocking, sorted-neighborhood, bigram indexing, canopy clustering and adaptive blocking [18,23,75] have been developed to address the problem of the quadratic time complexity of brute force comparison methods. The idea is to filter out obvious non-matches efficiently before executing the more detailed and time-consuming comparisons. In the following, we present a state-of-the-art framework that implements lossless instance matching based on a similar idea in detail.

5.4 The LIMES Algorithm

The original LIMES algorithm described in [111] addresses the scalability problem of link discovery by utilizing the *triangle inequality* in metric spaces to compute pessimistic estimates of instance similarities. Based on these approximations, LIMES can filter out a large number of instance pairs that cannot suffice the matching condition set by the user. The real similarities of the remaining instances pairs are then computed and the matching instances are returned.

Mathematical Framework In the remainder of this section, we use the following notations:

1. A is an affine space,
2. m, m_1, m_2, m_3 symbolize metrics on A,
3. x, y and z represent points from A and
4. α, β, γ and δ are scalars, i.e., elements of $\mathbb{R}$.

Definition 4 (Metric space). *A* metric space *is a pair* (A, m) *such that A is an affine space and* $m : A \times A \to \mathbb{R}$ *is a function such that for all x, y and z* $\in A$

1. $m(x, y) \geq 0$ (M_1) *(non-negativity),*
2. $m(x, y) = 0 \Leftrightarrow x = y$ (M_2) *(identity of indiscernibles),*
3. $m(x, y) = m(y, x)$ (M_3) *(symmetry) and*
4. $m(x, z) \leq m(x, y) + m(y, z)$ (M_4) *(*triangle inequality*).*

Note that the definition of a matching based on a similarity function σ can be rewritten for metrics m as follows:

Definition 5 (Instance Matching in Metric Spaces). *Given two sets S (source) and T (target) of instances, a metric* $m : S \times T \to [0, \infty[$ *and a threshold* $\theta \in [0, \infty[$*, the goal of instance matching task is to compute the set* $M = \{(s, t) | m(s, t) \leq \theta\}$.

Example of metrics on strings include the Levenshtein distance and the block distance. However, some popular measures such as JaroWinkler [160] do not satisfy the triangle inequality and are consequently not metrics. The rationale behind the LIMES framework is to make use of the boundary conditions entailed by the triangle inequality (TI) to reduce the number of comparisons (and thus the time complexity) necessary to complete a matching task. Given a metric space (A, m) and three points x, y and z in A, the TI entails that

$$m(x, y) \leq m(x, z) + m(z, y). \tag{1}$$

Without restriction of generality, the TI also entails that

$$m(x, z) \leq m(x, y) + m(y, z), \tag{2}$$

thus leading to the following boundary conditions in metric spaces:

$$m(x, y) - m(y, z) \leq m(x, z) \leq m(x, y) + m(y, z). \tag{3}$$

Inequality 3 has two major implications. The first is that the distance from a point x to any point z in a metric space can be approximated given the distance from x to a reference point y and the distance from the reference point y to z. Such a reference point is called an *exemplar* following [48]. The role of an exemplar is to be used as a sample of a portion of the metric space A. Given an input point x, knowing the distance from x to an exemplar y allows to compute lower and upper bounds of the distance from x to any other point z at a known distance from y. An example of such an approximation is shown in Figure 21. In this figure, all the points on the circle are subject to the same distance approximation. The distance from x to z is close to the lower bound of inequality 3, while the distance from x to z' is close to the upper bound of the same inequality.

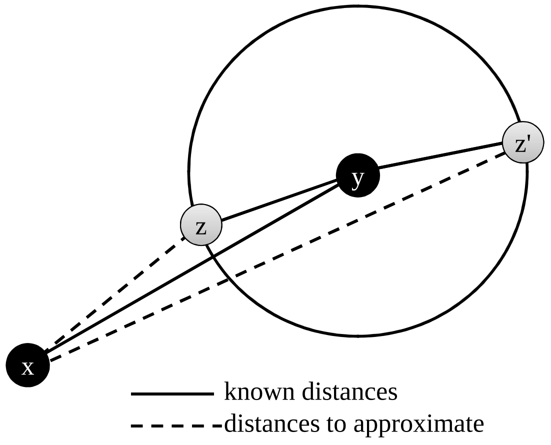


Fig. 16: Approximation of distances via exemplars. The lower bound of the distance from x to z can be approximated by $m(x, y) - m(y, z)$.

The second implication of inequality 3 is that the distance from x to z can only be smaller than θ if the lower bound of the approximation of the distance from x to z via *any exemplar y* is also smaller than θ. Thus, if the lower bound of the approximation of the distance $m(x, z)$ is larger than θ, then $m(x, z)$ itself must be larger than θ. Formally,

$$m(x, y) - m(y, z) > \theta \Rightarrow m(x, z) > \theta. \tag{4}$$

Supposing that all distances from instances $t \in T$ to exemplars are known, reducing the number of comparisons simply consists of using inequality 4 to compute an approximation of the distance from all $s \in S$ to all $t \in T$ and computing the real distance only for the (s, t) pairs for which the first term of inequality 4 does not hold. This is the core of the approach implemented by LIMES.

Computation of Exemplars The core idea underlying the computation of exemplars in LIMES is to select a set of exemplars in the metric space underlying the matching task in such a way that they are distributed uniformly in the metric space. One way to achieve this goal is by ensuring that the exemplars display a high dissimilarity. The approach used by LIMES to generate exemplars with this characteristic is shown in Algorithm 1.

Algorithm 1 Computation of Exemplars

```
Require: Number of exemplars $n$
Require: Target knowledge base $T$
  1. Pick random point $e_1 \in T$
  2. Set $E = E \cup \{e_1\}$;
  3. Compute the distance from $e_1$ to all $t \in T$
  while $|E| < n$ do
    4. Get a random point $e'$ such that $e' \in argmax_t \sum_{t \in T} \sum_{e \in E} m(t, e)$
    5. $E = E \cup \{e'\}$;
    6. Compute the distance from $e'$ to all $t \in T$
  end while
  7. Map each point in $t \in T$ to one of the exemplars $e \in E$ such that $m(t, e)$ is minimal
  return $E$
```

Let n be the desired number of exemplars and E the set of all exemplars. In step 1 and 2, LIMES initializes E by picking a random point e_1 in the metric space (T, m) and setting $E = \{e_1\}$. Then, it computes the similarity from the exemplar e_1 to every other point in T (step 3). As long as the size of E has not reached n, LIMES repeats steps 4 to 6: In step 4, a point $e' \in T$ such that the sum of the distances from e' to the exemplars $e \in E$ is maximal (there can be many of these points) is chosen randomly. This point is chosen as new exemplar and consequently added to E (step 5). Then, the distance from e' to all other points in T is computed (step 6). Once E has reached the size n, LIMES terminates the iteration. Finally, each point is mapped to the exemplar to which it is most similar (step 7) and the exemplar computation terminates (step 8). This algorithm has a constant time complexity of $O(|E||T|)$.

An example of the results of the exemplar computation algorithm ($|E| = 3$) is shown in Figure 17. The initial exemplar was the leftmost exemplar in the figure.

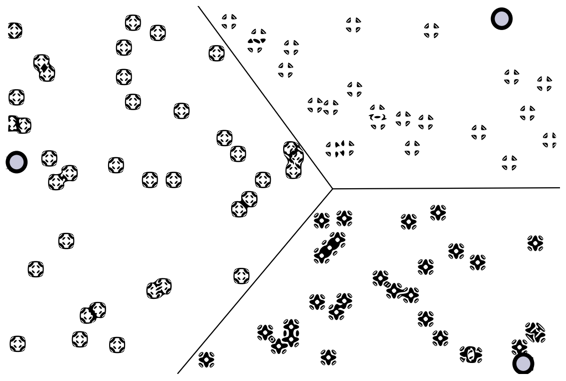


Fig. 17: Mapping of points to three exemplars in a metric space. The exemplars are displayed as gray disks.

Matching Based on Exemplars The instances associated with an exemplar $e \in E$ in step 7 of Algorithm 1 are stored in a list L_e sorted in descending order with respect to the distance to e. Let $\lambda_1^e ... \lambda_m^e$ be the elements of the list L_e. The goal of matching an instance s from a source knowledge base to a target knowledge base w.r.t. a metric m is to find all instances t of the target knowledge source such that $m(s,t) \leq \theta$, where θ is a given threshold. LIMES achieves this goal by using the matching algorithm based on exemplars shown in Algorithm 2.

LIMES only carries out a comparison when the approximation of the distance is less than the threshold. Moreover, it terminates the similarity computation for an exemplar e as soon as the first λ^e is found such that the lower bound of the distance is larger than θ. This break can be carried out because the list L_e is sorted, i.e., if $m(s,e) - m(e,\lambda_i^e) > \theta$ holds for the i^{th} element of L_e, then the same inequality holds for all $\lambda_j^e \in L_e$ with $j > i$. In the worst case, LIMES' matching algorithm has the time complexity $O(|S||T|)$, leading to a total worst time complexity of $O((|E| + |S|)|T|)$, which is larger than that of brute force approaches. However, as the results displayed in Figure 18 show, a correct parameterization of LIMES leads to significantly smaller numbers of comparisons and runtimes.

5.5 The $\mathcal{HR}^3$ algorithm

Let S resp. T be the source and target of a Link Discovery task. One of the key ideas behind time-efficient Link Discovery algorithms $\mathcal{A}$ is to reduce the number of comparisons that are effectively carried out to a number $C(\mathcal{A}) < |S||T|$. The reduction ratio RR

Algorithm 2 LIMES' Matching algorithm

```
Require: Set of exemplars $E$
Require: Instance $s \in S$
Require: Metric m
Require: threshold $\theta$
  1. $M = \emptyset$
  for $e \in |E|$ do
    if $m(s, e) \leq \theta$ then
      2. $M = M \cup \{e\}$
      for $i = 1...|L_e|$ do
        if $(m(s, e) - m(e, \lambda_i^e)) \leq \theta$ then
          if $m(s, \lambda_i^e) \leq \theta$ then
            3. $M = M \cup \{(s, \lambda_i^e\})$
          end if
        else
          break
        end if
      end for
    end if
  end for
  return $M$
```

of an algorithm $\mathcal{A}$ is given by

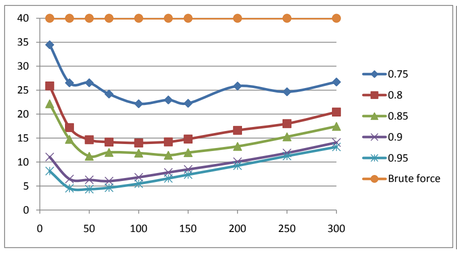
$$RR(\mathcal{A}) = 1 - \frac{C(\mathcal{A})}{|S||T|}. \tag{5}$$

$RR(\mathcal{A})$ captures how much of the Cartesian product $|S||T|$ was not explored before the output of $\mathcal{A}$ was reached. It is obvious that even an optimal lossless solution which performs only the necessary comparisons cannot achieve a RR of 1. Let C_{min} be the minimal number of comparisons necessary to complete the Link Discovery task without losing recall, i.e., $C_{min} = |\mathcal{M}|$. The relative reduction ratio $RRR(\mathcal{A})$ is defined as the portion of the minimal number of comparisons that was carried out by the algorithm $\mathcal{A}$ before it terminated. Formally

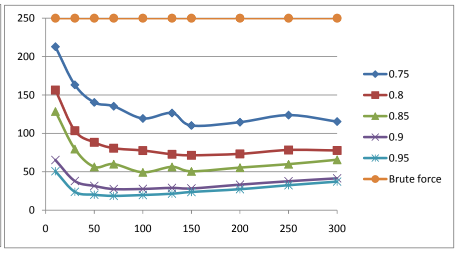
$$RRR(\mathcal{A}) = \frac{1 - \frac{C_{min}}{|S||T|}}{1 - \frac{C(\mathcal{A})}{|S||T|}} = \frac{|S||T| - C_{min}}{|S||T| - C(\mathcal{A})}. \tag{6}$$

$RRR(\mathcal{A})$ indicates how close $\mathcal{A}$ is to the optimal solution with respect to the number of candidates it tests. Given that $C(\mathcal{A}) \geq C_{min}$, $RRR(\mathcal{A}) \geq 1$. Note that the larger the value of $RRR(\mathcal{A})$, the poorer the performance of $\mathcal{A}$ with respect to the task at hand.

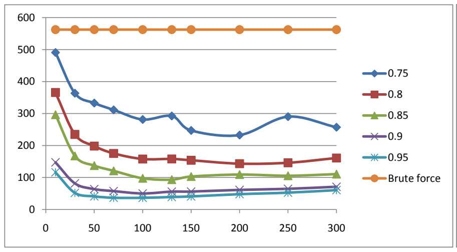
The main observation that led $\mathcal{HR}^3$ is that while most algorithms aim to optimize their RR (and consequently their RRR), most approaches do not provide any guarantee with respect to the RR (and consequently the RRR) that they can achieve. The approach is the first mathematically optimal algorithm w.r.t the reduction ratio that it can achieve, i.e., the first approach such that given any relative reduction ratio r, there is always a setting that leads $\mathcal{HR}^3$ achieving a relative reduction ratio r' with $r' \leq r$. To achieve this goal $\mathcal{HR}^3$ relies on space tiling as introduced by the HYPPO algorithm [108].



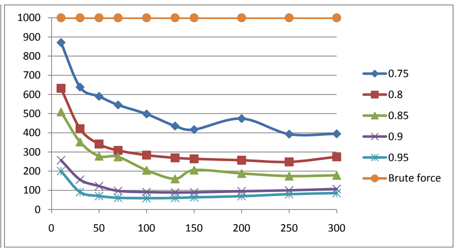
(a) Size = 2000



(b) Size = 5000



(c) Size = 7500



(d) Size = 10000

Fig. 18: Comparisons required by LIMES for different numbers of exemplars on knowledge bases of different sizes. The x-axis shows the number of exemplars, the y-axis the number of comparisons in multiples of 10^5.

Space Tiling for Link Discovery HYPPO addresses the problem of efficiently mapping instance pairs $(s,t) \in S \times T$ described by using exclusively numeric values in a n-dimensional metric space and has been shown to outperform the state of the art in previous work [108]. The observation behind space tiling is that in spaces (Ω, δ) with orthogonal, (i.e., uncorrelated) dimensions[17], common metrics for Link Discovery can be decomposed into the combination of functions $\phi_{i,i\in\{1...n\}}$ which operate on exactly one dimension of $\Omega : \delta = f(\phi_1, ..., \phi_n)$. For Minkowski distances of order p, $\phi_i(x,\omega) = |x_i - \omega_i|$ for all values of i and $\delta(x,\omega) = \sqrt[p]{\sum\limits_{i=1}^{n} \phi_i^p(x,\omega)^p}$. A direct consequence of this observation is the inequality $\phi_i(x,\omega) \le \delta(x,\omega)$. The basic insight that results this observation is that the hypersphere $H(\omega,\theta) = \{x \in \Omega : \delta(x,\omega) \le \theta\}$ is a subset of the hypercube V defined as $V(\omega,\theta) = \{x \in \Omega : \forall i \in \{1...n\}, \phi_i(x_i,\omega_i) \le \theta\}$. Consequently, one can reduce the number of comparisons necessary to detect all elements of $H(\omega,\theta)$ by discarding all elements which are not in $V(\omega,\theta)$ as non-matches. Let $\Delta = \theta/\alpha$, where $\alpha \in \mathbb{N}$ is the *granularity parameter* that controls how fine-grained the space tiling should be (see Figure 19 for an example). We first tile Ω into the adjacent hypercubes (short: cubes) C

[17] Note that in all cases, a space transformation exists that can map a space with correlated dimensions to a space with uncorrelated dimensions.

that contain all the points ω such that

$$\forall i \in \{1...n\}, c_i\Delta \leq \omega_i < (c_i + 1)\Delta \text{ with } (c_1, ..., c_n) \in \mathbb{N}^n. \quad (7)$$

We call the vector $(c_1, ..., c_n)$ the coordinates of the cube C. Each point $\omega \in \Omega$ lies in the cube $C(\omega)$ with coordinates $(\lfloor \omega_i/\Delta \rfloor)_{i=1...n}$. Given such a space tiling, it is obvious that $V(\omega, \theta)$ consists of the union of the cubes such that $\forall i \in \{1...n\} : |c_i - c(\omega)_i| \leq \alpha$.

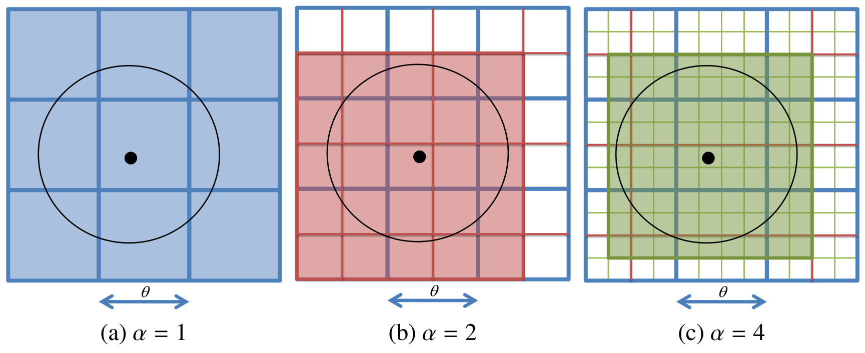


Fig. 19: Space tiling for different values of α. The colored squares show the set of elements that must be compared with the instance located at the black dot. The points within the circle lie within the distance θ of the black dot. Note that higher values of α lead to a better approximation of the hypersphere but also to more hypercubes.

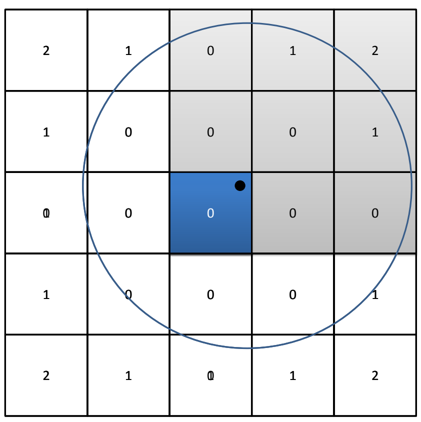
$\mathcal{HR}^3$'s indexing scheme Let $\omega \in \Omega = S \cup T$ be an arbitrary reference point. Furthermore, let δ be the Minkowski distance of order p. The *index* function is defined as follows:

$$index(C, \omega) = \begin{cases} 0 \text{ if } \exists i : |c_i - c(\omega)_i| \leq 1 \text{ with } i \in \{1, ..., n\}, \\ \sum_{i=1}^{n} (|c_i - c(\omega)_i| - 1)^p \text{ else,} \end{cases} \quad (8)$$

where C is a hypercube resulting from a space tiling and $\omega \in \Omega$. Figure 20 shows an example of such indexes for $p = 2$ with $\alpha = 2$ (Figure 20a) and $\alpha = 4$ (Figure 20b). Note that the blue square with index 0 contains the reference point ω. All elements of C must only be compared with the elements of cubes C' such that $index(C, C') \leq \alpha^p$. The authors of [109] prove formally that given this approach to space tiling, the following theorem holds:

Theorem 1. $\lim_{\alpha \to \infty} RRR(\mathcal{HR}^3, \alpha) = 1.$

This conclusion is illustrated by Figure 21, which shows the space tiling computed by $\mathcal{HR}^3$ for different values of α with $p = 2$ and $n = 2$. The higher α, the closer the approximation is to a circle. Note that these results allow to conclude that for any *RRR*-value r larger than 1, there is a setting of $\mathcal{HR}^3$ that can compute links with a *RRR* smaller or equal to r.


(a) $\alpha = 2$

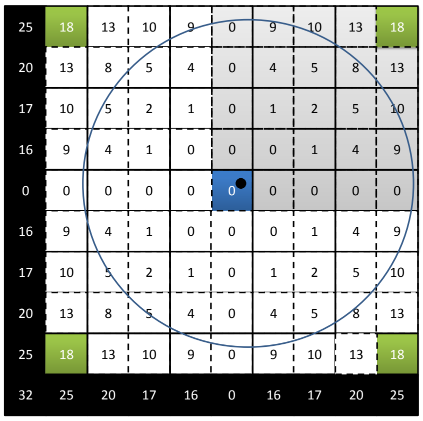

(b) $\alpha = 4$

Fig. 20: Space tiling and resulting index for a two-dimensional example. Note that the index in both subfigures was generated for exactly the same portion of space. The black dot stands for the position of ω.

Evaluation $\mathcal{HR}^3$ was evaluated against HYPPO w.r.t. to the number of comparisons that it has to carry out in several settings. In the first and second experiments, the goal was to deduplicate DBpedia places by comparing their names (`rdfs:label`), minimum elevation, elevation and maximum elevation. 2988 entities possessed all four properties. The Euclidean metric was applied to the last three values with the thresholds 49 meters resp. 99 meters for the first resp. second experiment. The third and fourth experiments aimed to discover links between Geonames and LinkedGeoData. This experiment was of considerably larger scale than the first one, as we compared 74458 entities in Geonames with 50031 entities from LinkedGeoData. Again, the number of comparisons necessary to complete the task by using the Euclidean metric was measured. The distance thresholds were set to 1 resp. 9° in experiment 3 resp. 4. We ran all experiments on the same Windows 7 Enterprise 64-bit computer with a 2.8GHz i7 processor with 8GB RAM. The JVM was allocated 7GB RAM to ensure that the runtimes were not influenced by swapping. Only one of the kernels of the processors was used.

The results (see Figure 22) show that $\mathcal{HR}^3$ can reduce the overhead in comparisons (i.e., the number of unnecessary comparisons divided by the number of necessary comparisons) from approximately 24% for HYPPO to approximately 6% (granularity = 32). In experiment 2, the overhead is reduced from 4.1% to 2%. This difference in overhead reduction is mainly due to the data clustering around certain values and the clusters having a radius between 49 meters and 99 meters. Thus, running the algorithms with a threshold of 99 meters led to only a small a-priori overhead and HYPPO performing remarkably well. Still, even on such data distributions, $\mathcal{HR}^3$ was able to discard even more data and to reduce the number of unnecessary computations by more than 50% relative. In the best case (Exp. 4, $\alpha = 32$, see Figure 22d), $\mathcal{HR}^3$ required approximately

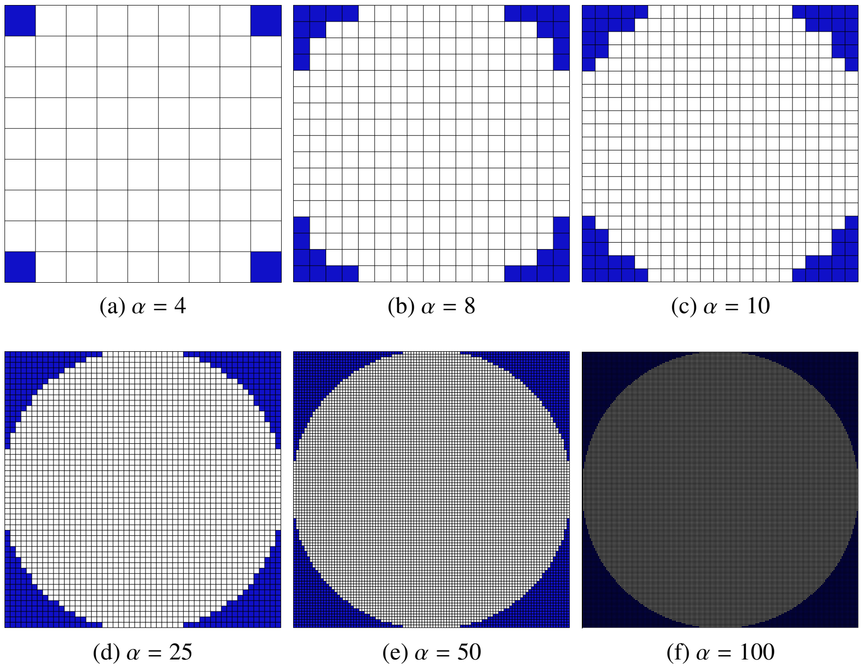


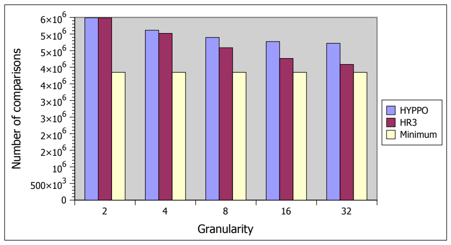
Fig. 21: Space tilings generated by $\mathcal{HR}^3$ for different values of α. The white squares are selected for comparisons with the elements of the square in the middle of the figure whilst the colored ones are discarded.

4.13×10^6 less comparisons than HYPPO for $\alpha = 32$. Even for the smallest setting (Exp. 1, see Figure 22a), $\mathcal{HR}^3$ still required 0.64×10^6 less comparisons.

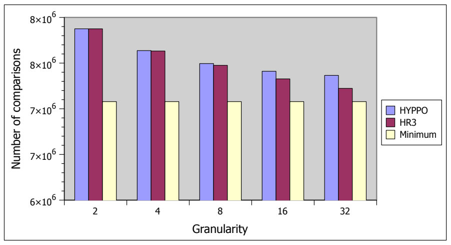
5.6 Active Learning of Link Specifications

The second challenge of Link Discovery is the time-efficient discovery of link specifications for a particular linking task. Several approaches have been proposed to achieve this goal, of which most rely on genetic programming [69,116,114]. The COALA (Correlation-Aware Active Learning) approach was implemented on top of the genetic programming approach EAGLE [115] with the aim of improving the selection of positive and negative examples during active learning. In the following, we give an overview of COALA.

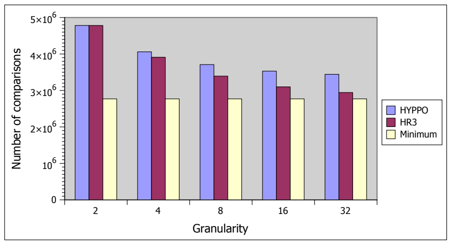
Intuition Let $\mathcal{N}$ be the set of most informative negative and $\mathcal{P}$ the set of most informative negative examples w.r.t. an informativeness function ifm (e.g., the distance from the decision boundary).used by a curious classifier [140] The basic insight behind COALA is that the correlation between the features of the elements of $\mathcal{N}$ and $\mathcal{P}$ should play a



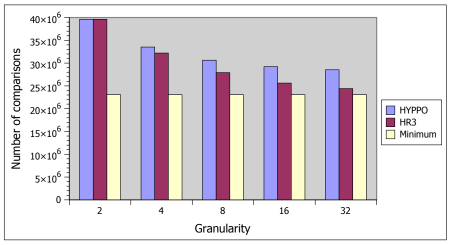
(a) Experiment 1



(b) Experiment 2



(c) Experiment 3

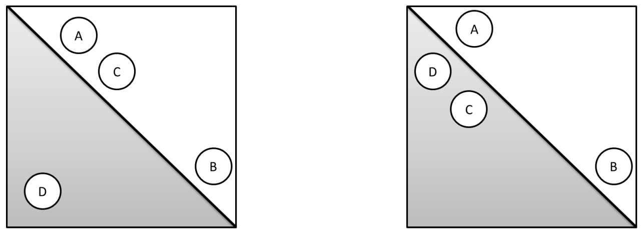


(d) Experiment 4

Fig. 22: Number of comparisons for $\mathcal{HR}^3$ and HYPPO.

role when computing the sets $\mathcal{I}^+$ and $\mathcal{I}^-$ of positive resp. negative queries for the oracle. In particular, two main factors affect the information content of a link candidate: its similarity to elements of its presumed class and to elements of the other class. For the sake of simplicity, we will assume that the presumed class of the link candidate of interest is $+1$, i.e., that the link candidate was classified as positive by the current curious classifier. Our insights yet hold symmetrically for link candidates whose presumed class is -1.

Let $A = (s_A, t_A), B = (s_B, t_B) \in \mathcal{P}$ to be two link candidates which are equidistant from C's boundary. Consider Figure 23a, where $\mathcal{P} = \{A, B, C\}$ and $\mathcal{N} = \{D\}$. The link candidate B is on on average most distant from any other elements of $\mathcal{P}$. Thus, it is more likely to be a statistical outlier than A. Hence, making a classification error on B should not have the same impact as an erroneous classification of link candidate A, which is close to another presumably positive link candidate, C. Consequently, B should be considered less informative than A. Approaches that make use of this information are said to exploit the *intra-class correlation*. Now, consider Figure 23b, where $\mathcal{P} = \{A, B\}$ and $\mathcal{N} = \{C, D\}$. While the probability of A being an outlier is the same as B's, A is still to be considered more informative than B as it is located closer to elements of $\mathcal{N}$ and can thus provide more information on where to set the classifier boundary. This information is dubbed *inter-class correlation*. Several approaches that make use of these two types of correlations can be envisaged. In the following, we present two approaches for these purposes. The first makes use of intra-class correlations and relies on graph clustering. The second approach relies on the spreading activation principle in combination with weight decay. We assume that the complex similarity function σ



(a) Intra-correlation (b) Inter-correlation

Fig. 23: Examples of correlations within classes and between classes. In each subfigure, the gray surface represent $\mathcal{N}$ while the white surface stands for $\mathcal{P}$. The oblique line is C's boundary.

underlying C is computed by combining n atomic similarity functions $\sigma_1, \ldots, \sigma_n$. This combination is most commonly carried out by using metric operators such as min, max or linear combinations.[18] Consequently, each link candidate (s, t) can be described by a vector $(\sigma_1(s, t), \ldots, \sigma_n(s, t)) \in [0, 1]^n$. We define the *similarity of link candidates* $sim : (S \times T)^2 \to [0, 1]$ to be the inverse of the Euclidean distance in the space spawned by the similarities σ_1 to σ_n. Hence, the similarity of two link candidates (s, t) and (s', t') is given by:

$$sim((s, t), (s', t')) = \frac{1}{1 + \sqrt{\sum\limits_{i=1}^{n} (\sigma_i(s, t) - \sigma_i(s', t'))^2}}. \tag{9}$$

Note that we added 1 to the denominator to prevent divisions by 0.

Graph Clustering The basic intuition behind using clustering for COALA is that groups of very similar link candidates can be represented by a single link candidate. Consequently, once a representative of a group has been chosen, all other elements of the group become less informative. An example that illustrates this intuition is given in Figure 24. We implemented COALA based on clustering as follows: In each iteration, we begin by first selecting two sets $\mathcal{S}^+ \subseteq \mathcal{P}$ resp. $\mathcal{S}^- \subseteq \mathcal{N}$ that contain the positive resp. negative link candidates that are most informative for the classifier at hand. Formally, $\mathcal{S}^+$ fulfills

$$\forall x \in \mathcal{S}^+ \; \forall y \in \mathcal{P}, y \notin \mathcal{S}^+ \to \text{ifm}(y) \leq \text{ifm}(x). \tag{10}$$

The analogous equation holds for $\mathcal{S}^-$. In the following, we will explain the further steps of the algorithm for $\mathcal{S}^+$. The same steps are carried out for $\mathcal{S}^-$.

First, we compute the similarity of all elements of $\mathcal{S}^+$ by using the similarity function shown in Equation 9. In the resulting similarity matrix, we set all elements of the diagonal to 0. Then, for each $x \in \mathcal{S}^+$, we only retain a fixed number ec of highest similarity values and set all others to 0. The resulting similarity matrix is regarded as the adjacency matrix of an undirected weighted graph $G = (V, E, sim)$. G's set of nodes V

[18] See [110] for a more complete description of a grammar for link specifications.

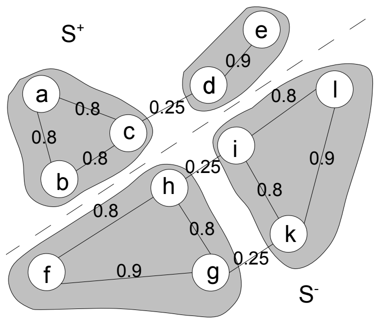


Fig. 24: Example of clustering. One of the most informative single link candidate is selected from each cluster. For example, d is selected from the cluster $\{d, e\}$.

is equal to $\mathcal{S}^+$. The set of edges E is a set of 2-sets[19] of link candidates. Finally, the weighted function is the similarity function *sim*. Note that *ec* is the minimal degree of nodes in G.

In a second step, we use the graph G as input for a graph clustering approach. The resulting clustering is assumed to be a partition $\mathcal{V}$ of the set V of vertices of G. The informativeness of partition $V_i \in \mathcal{V}$ is set to $\max_{x \in V_i} \text{ifm}(x)$. The final step of our approach consists of selecting the most informative node from each of the k most informative partitions. These are merged to generate I^+, which is sent as query to the oracle. The computation of $I^-$is carried out analogously. Note that this approach is generic in the sense that it can be combined with any graph clustering algorithm that can process weighted graphs as well as with any informativeness function ifm. Here, we use BorderFlow [117] as clustering algorithm because (1) it has been used successfully in several other applications such as the creation of SPARQL benchmarks [101] and the analysis of protein-protein interactions [107]. and (2) it is parameter-free and does not require any tuning.

Spreading Activation with Weight Decay The idea behind spreading activation with weight decay (WD) is to combine the intra- and inter-class correlation to determine the informativeness of each link candidate. Here, we begin by computing the set $\mathcal{S} = \mathcal{S}^+ \cup \mathcal{S}^-$, where $\mathcal{S}^+$ and $\mathcal{S}^-$ are described as above. Let $\mathfrak{s}_i$ and $\mathfrak{s}_j$ be the i^{th} and j^{th} elements of $\mathcal{S}$. We then compute the quadratic similarity matrix $\mathcal{M}$ with entries $m_{ij} = sim(\mathfrak{s}_i, \mathfrak{s}_j)$ for $i \neq j$ and 0 else. Note that both negative and positive link candidates belong to $\mathcal{S}$. Thus, $\mathcal{M}$ encodes both inter- and intra-class correlation. In addition to $\mathcal{M}$, we compute the activation vector $\mathcal{A}$ by setting its entries to a_i =ifm($\mathfrak{s}_i$). In the following, $\mathcal{A}$ is considered to be a column vector.

In a first step, we normalize the activation vector $\mathcal{A}$ to ensure that the values contained therein do not grow indefinitely. Then, in a second step, we set $\mathcal{A} = \mathcal{A} + \mathcal{M} \times \mathcal{A}$. This has the effect of propagating the activation of each $\mathfrak{s}$ to all its neighbors according to the weights of the edges between $\mathfrak{s}$ and its neighbors. Note that elements of $\mathcal{S}^+$ that

[19] A n-set is a set of magnitude n.

are close to elements of $\mathcal{S}^-$ get a higher activation than elements of $\mathcal{S}^+$ that are further away from $\mathcal{S}^-$ and vice-versa. Moreover, elements at the center of node clusters (i.e., elements that are probably no statistical outliers) also get a higher activation than elements that are probably outliers. The idea behind the weight decay step is to update the matrix by setting each m_{ij} to m_{ij}^r, where $r > 1$ is a fix exponent. This is the third step of the algorithm. Given that $\forall i \forall j\ m_{ij} \leq 1$, the entries in the matrix get smaller with time. By these means, the amount of activation transferred across long paths is reduced. We run this three-step procedure iteratively until all non-1 entries of the matrix are less or equal to a threshold $\epsilon = 10^{-2}$. The k elements of $\mathcal{S}^+$ resp. $\mathcal{S}^-$ with maximal activation are returned as I^+ resp. I^-. In the example shown in Figure 25, while all nodes from $\mathcal{S}^+$ and $\mathcal{S}^-$ start with the same activation, two nodes get the highest activation after only 3 iterations.

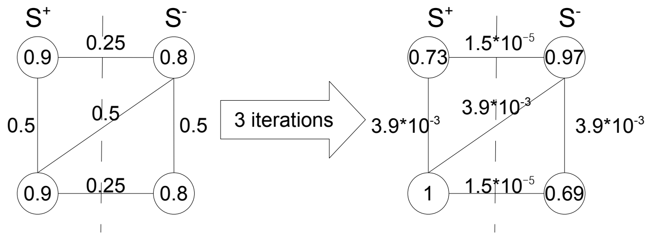


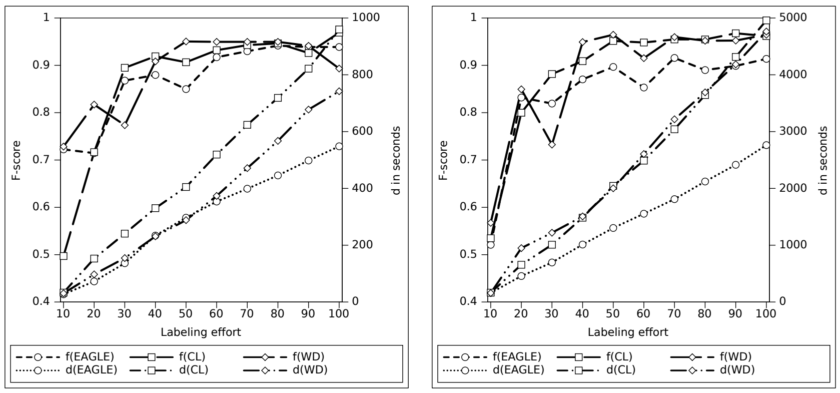
Fig. 25: Example of weight decay. Here r was set to 2. The left picture shows the initial activations and similarity scores while the right picture shows the results after 3 iterations. Note that for the sake of completeness the weights of the edges were not set to 0 when they reached ϵ.

Evaluation COALA was evaluated by running weight decay and clustering in combination with EAGLE, an active genetic programming approach for learning link specifications. Throughout the following experiments, EAGLE's mutation and crossover rates were set to 0.6. Individuals were given a 70% chance to get selected for reproduction. The population sizes were set to 20 and 100. We set $k = 5$ and ran our experiments for 10 iterations. Between each iteration we evolved the populations for 50 generations. We ran our experiments on two real-world datasets and three synthetic datasets. The synthetic datasets consisted of the Persons1, Person2 and Restaurants datasets from the OAEI 2010 benchmark[20]. The real-world datasets consisted of the ACM-DBLP and Abt-Buy datasets, which were extracted from websites or databases [75] [21]. Given that genetic programming is non-deterministic, all results presented below are the means

[20] http://oaei.ontologymatching.org/2010/

[21] http://dbs.uni-leipzig.de/en/research/projects/object_matching/fever/benchmark_datasets_for_entity_resolution

of 5 runs. Each experiment was ran on a single thread of a server running JDK1.7 on Ubuntu 10.0.4 and was allocated maximally 2GB of RAM. The processors were 2.0GHz Quadcore AMD Opterons. An excerpt of the results is shown in Figure 26. While the results show that COALA outperform EAGLE, it remains unclear whether WD or CL is the best approach to achieving a faster convergence towards the optimal solution.



(a) Population = 20 individuals. (b) Population = 100 individuals.

Fig. 26: F-score and runtime on the ACM-DBLP dataset. f(X) stands for the F-score achieved by algorithm X, while d(X) stands for the total duration required by the algorithm.

5.7 Conclusion

We presented and discussed linking approaches for Linked Data and the challenges they face. In addition, we gave an overview of several state-of-the-art approaches for instance matching for Linked Data. We then presented time-efficient approaches for link discovery. Finally, we presented a state-of-the-art approach for the active learning of link specifications. This approach can be easily extended to learn specifications automatically. [22] Novel challenges that need to be addressed include the automatic management of resources for link specifications. First works on running link discovery in parallel have shown that using massively parallel hardware such as GPUs can lead to better results that using cloud implementations even on considerably large datasets [113]. Detecting the right resources for linking automatically given a hardware landscape is yet still a dream to achieve.

[22] Such an extension is the basis of the self-configuration algorithm of the SAIM framework.

6 Enrichment

The term *enrichment* in this chapter refers to the (semi-)automatic extension of a knowledge base schema. It describes the process of increasing the expressiveness and semantic richness of a knowledge base. Usually, this is achieved by adding or refining terminological axioms.

Enrichment methods can typically be applied in a *grass-roots* approach to knowledge base creation. In such an approach, the whole ontological structure is not created upfront, but evolves with the data in a knowledge base. Ideally, this enables a more agile development of knowledge bases. In particular, in the context of the Web of Linked Data such an approach appears to be an interesting alternative to more traditional ontology engineering methods. Amongst others, Tim Berners-Lee advocates to get "raw data now"[23] and worry about the more complex issues later.

Knowledge base enrichment can be seen as a sub-discipline of ontology learning. Ontology learning is more general in that it can rely on external sources, e.g. written text, to create an ontology. The term knowledge base enrichment is typically used when already existing data in the knowledge base is analysed to improve its schema.

Enrichment methods span several research areas like knowledge representation and reasoning, machine learning, statistics, natural language processing, formal concept analysis and game playing. Considering the variety of methods, we structure this section as follows: First, we give an overview of different types of enrichment and list some typical methods and give pointers to references, which allow the reader to obtain more information on a topic. In the second part, we describe a specific software – the ORE tool – in more detail.

6.1 State of the Art and Types of Enrichment

Ontology enrichment usually involves applying heuristics or machine learning techniques to find axioms, which can be added to an existing ontology. Naturally, different techniques have been applied depending on the specific type of axiom.

One of the most complex tasks in ontology enrichment is to find *definitions* of classes. This is strongly related to Inductive Logic Programming (ILP) [120] and more specifically supervised learning in description logics. Research in those fields has many applications apart from being applied to enrich ontologies. For instance, it is used in the life sciences to detect whether drugs are likely to be efficient for particular diseases. Work on learning in description logics goes back to e.g. [33,34], which used so-called *least common subsumers* to solve the learning problem (a modified variant of the problem defined in this article). Later, [17] invented a refinement operator for $\mathcal{ALER}$ and proposed to solve the problem by using a top-down approach. [39,65,66] combine both techniques and implement them in the YINYANG tool. However, those algorithms tend to produce very long and hard-to-understand class expressions. The algorithms implemented in DL-Learner [85,86,77,87,79] overcome this problem and investigate the learning problem and the use of top down refinement in detail. DL-FOIL [42] is a similar approach, which is based on a mixture of upward and downward refinement

[23] `http://www.ted.com/talks/tim_berners_lee_on_the_next_web.html`

of class expressions. They use alternative measures in their evaluation, which take the open world assumption into account, which was not done in ILP previously. Most recently, [81] implements appropriate heuristics and adaptations for learning definitions in ontologies. The focus in this work is efficiency and practical application of learning methods. The article presents plugins for two ontology editors (Protégé and OntoWiki) as well stochastic methods, which improve previous methods by an order of magnitude. For this reason, we will analyse it in more detail in the next subsection. The algorithms presented in the article can also learn *super class axioms*.

A different approach to learning the definition of a named class is to compute the so called *most specific concept* (msc) for all instances of the class. The most specific concept of an individual is the most specific class expression, such that the individual is instance of the expression. One can then compute the *least common subsumer* (lcs) [16] of those expressions to obtain a description of the named class. However, in expressive description logics, an msc does not need to exist and the lcs is simply the disjunction of all expressions. For light-weight logics, such as $\mathcal{EL}$, the approach appears to be promising.

Other approaches, e.g. [90] focus on learning in hybrid knowledge bases combining ontologies and *rules*. Ontology evolution [91] has been discussed in this context. Usually, hybrid approaches are a generalisation of concept learning methods, which enable powerful rules at the cost of efficiency (because of the larger search space). Similar as in knowledge representation, the tradeoff between expressiveness of the target language and efficiency of learning algorithms is a critical choice in symbolic machine learning.

Another enrichment task is *knowlege base completion*. The goal of such a task is to make the knowledge base complete in a particular well-defined sense. For instance, a goal could be to ensure that all subclass relationships between named classes can be inferred. The line of work starting in [132] and further pursued in e.g. [15] investigates the use of *formal concept analysis* for completing knowledge bases. It is promising, although it may not be able to handle noise as well as a machine learning technique. A Protégé plugin [139] is available. [153] proposes to improve knowledge bases through relational exploration and implemented it in the *RELExO framework*[24]. It focuses on simple relationships and the knowledge engineer is asked a series of questions. The knowledge engineer either must positively answer the question or provide a counterexample.

[154] focuses on learning *disjointness* between classes in an ontology to allow for more powerful reasoning and consistency checking. To achieve this, it can use the ontology itself, but also texts, e.g. Wikipedia articles corresponding to a concept. The article includes an extensive study, which shows that proper modelling disjointness is actually a difficult task, which can be simplified via this ontology enrichment method.

Another type of ontology enrichment is schema mapping. This task has been widely studied and will not be discussed in depth within this chapter. Instead, we refer to [31] for a survey on ontology mapping.

There are further more light-weight ontology enrichment methods. For instance, *taxonomies* can be learned from simple tag structures via heuristics [29,152]. Similarly, “properties of properties” can be derived via simple statistical analysis. This includes

[24] `http://code.google.com/p/relexo/`

the detection whether a particular property might be symmetric, function, reflexive, inverse functional etc. Similarly, domains and ranges of properties can be determined from existing data. Enriching the schema with domain and range axioms allows to find cases, where properties are misused via OWL reasoning.

Type/Aim	**References**
Taxonomies	[162,29,152]
Definitions	often done via ILP approaches such as [85,86,87,81,42,39,65,66,17], genetic approaches [77] have also been used
Super Class Axioms	[81,152,29]
Rules in Ontologies	[90,91]
Disjointness	[154]
Properties of properties	[29,45]
Alignment	challenges: [141], recent survey: [31]
Completion	formal concept analysis and relational exploration [15,153,139]

Table 3: Work in ontology enrichment grouped by type or aim of learned structures.

In the following subsection, we describe an enrichment approach for learning definitions and super class axioms in more detail. The algorithm was recently developed by the first authors and is described in full detail in [81].

6.2 Class Expression Learning in DL-Learner

The Semantic Web has recently seen a rise in the availability and usage of knowledge bases, as can be observed within the Linking Open Data Initiative, the TONES and Protégé ontology repositories, or the Watson search engine. Despite this growth, there is still a lack of knowledge bases that consist of sophisticated schema information and instance data adhering to this schema. Several knowledge bases, e.g. in the life sciences, only consist of schema information, while others are, to a large extent, a collection of facts without a clear structure, e.g. information extracted from data bases or texts. The combination of sophisticated schema and instance data allows powerful reasoning, consistency checking, and improved querying possibilities. We argue that being able to learn OWL class expressions[25] is a step towards achieving this goal.

Example 1. As an example, consider a knowledge base containing a class `Capital` and instances of this class, e.g. London, Paris, Washington, Canberra etc. A machine learning algorithm could, then, suggest that the class `Capital` may be equivalent to one of the following OWL class expressions in Manchester OWL syntax[26]:

```
    City and isCapitalOf at least one GeopoliticalRegion
    City and isCapitalOf at least one Country
```

[25] `http://www.w3.org/TR/owl2-syntax/#Class_Expressions`

[26] For details on Manchester OWL syntax (e.g. used in Protégé, OntoWiki) see [62].

Both suggestions could be plausible: The first one is more general and includes cities that are capitals of states, whereas the latter one is stricter and limits the instances to capitals of countries. A knowledge engineer can decide which one is more appropriate, i.e. a semi-automatic approach is used, and the machine learning algorithm should guide her by pointing out which one fits the existing instances better. Assuming the knowledge engineer decides for the latter, an algorithm can show her whether there are instances of the class `Capital` which are neither instances of `City` nor related via the property `isCapitalOf` to an instance of `Country`.[27] The knowledge engineer can then continue to look at those instances and assign them to a different class as well as provide more complete information; thus improving the quality of the knowledge base. After adding the definition of `Capital`, an OWL reasoner can compute further instances of the class which have not been explicitly assigned before.

Using machine learning for the generation of suggestions instead of entering them manually has the advantage that 1.) the given suggestions fit the instance data, i.e. schema and instances are developed in concordance, and 2.) the entrance barrier for knowledge engineers is significantly lower, since understanding an OWL class expression is easier than analysing the structure of the knowledge base and creating a class expression manually. Disadvantages of the approach are the dependency on the availability of instance data in the knowledge base and requirements on the quality of the ontology, i.e. modelling errors in the ontology can reduce the quality of results.

Overall, we describe the following in this chapter:

– extension of an existing learning algorithm for learning class expressions to the ontology engineering scenario,
– presentation and evaluation of different heuristics,
– showcase how the enhanced ontology engineering process can be supported with plugins for Protégé and OntoWiki,
– evaluation of the presented algorithm with several real ontologies from various domains.

The adapted algorithm for solving the learning problems, which occur in the ontology engineering process, is called *CELOE (Class Expression Learning for Ontology Engineering)*. It was implemented within the open-source framework DL-Learner.[28] DL-Learner [78,79] leverages a modular architecture, which allows to define different types of components: knowledge sources (e.g. OWL files), reasoners (e.g. DIG[29] or OWL API based), learning problems, and learning algorithms. In this overview, we focus on the latter two component types, i.e. we define the class expression learning problem in ontology engineering and provide an algorithm for solving it.

Learning Problem The process of learning in logics, i.e. trying to find high-level explanations for given data, is also called *inductive reasoning* as opposed to *inference* or *deductive reasoning*. The main difference is that in deductive reasoning it is formally

[27] This is not an inconsistency under the standard OWL open world assumption, but rather a hint towards a potential modelling error.
[28] `http://dl-learner.org`
[29] `http://dl.kr.org/dig/`

shown whether a statement follows from a knowledge base, whereas in inductive learning new statements are invented. Learning problems, which are similar to the one we will analyse, have been investigated in *Inductive Logic Programming* [120] and, in fact, the method presented here can be used to solve a variety of machine learning tasks apart from ontology engineering.

In the ontology learning problem we consider, we want to learn a formal description of a class A, which has (inferred or asserted) instances in the considered ontology. In the case that A is already described by a class expression C via axioms of the form $A \sqsubseteq C$ or $A \equiv C$, those can be either refined, i.e. specialised/generalised, or relearned from scratch by the learning algorithm. To define the class learning problem, we need the notion of a *retrieval* reasoner operation $R_{\mathcal{K}}(C)$. $R_{\mathcal{K}}(C)$ returns the set of all instances of C in a knowledge base $\mathcal{K}$. If $\mathcal{K}$ is clear from the context, the subscript can be omitted.

Definition 6 (class learning problem). *Let an existing named class A in a knowledge base $\mathcal{K}$ be given. The* class learning problem *is to find an expression C such that $R_{\mathcal{K}}(C) = R_{\mathcal{K}}(A)$.*

Clearly, the learned expression C is a description of (the instances of) A. Such an expression is a candidate for adding an axiom of the form $A \equiv C$ or $A \sqsubseteq C$ to the knowledge base $\mathcal{K}$. If a solution of the learning problem exists, then the used base learning algorithm (as presented in the following subsection) is complete, i.e. guaranteed to find a correct solution if one exists in the target language and there are no time and memory constraints (see [86,87] for the proof). In most cases, we will not find a solution to the learning problem, but rather an approximation. This is natural, since a knowledge base may contain false class assignments or some objects in the knowledge base are described at different levels of detail. For instance, in Example 1, the city "Apia" might be typed as "Capital" in a knowledge base, but not related to the country "Samoa". However, if most of the other cities are related to countries via a role `isCapitalOf`, then the learning algorithm may still suggest `City and isCapitalOf at least one Country` since this describes the majority of capitals in the knowledge base well. If the knowledge engineer agrees with such a definition, then a tool can assist him in completing missing information about some capitals.

According to Occam's razor [26] simple solutions of the learning problem are to be preferred over more complex ones, because they are more readable. This is even more important in the ontology engineering context, where it is essential to suggest simple expressions to the knowledge engineer. We measure simplicity as the *length* of an expression, which is defined in a straightforward way, namely as the sum of the numbers of concept, role, quantifier, and connective symbols occurring in the expression. The algorithm is biased towards shorter expressions. Also note that, for simplicity the definition of the learning problem itself does enforce coverage, but not prediction, i.e. correct classification of objects which are added to the knowledge base in the future. Concepts with high coverage and poor prediction are said to *overfit* the data. However, due to the strong bias towards short expressions this problem occurs empirically rarely in description logics [87].

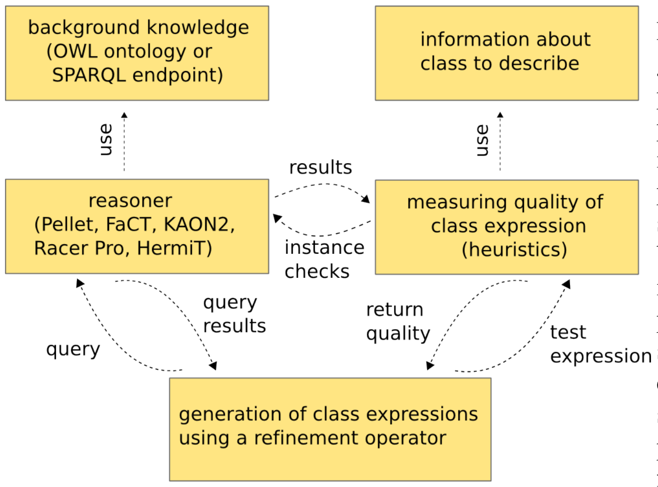

background knowledge (OWL ontology or SPARQL endpoint)
information about class to describe
use
use
results
reasoner (Pellet, FaCT, KAON2, Racer Pro, HermiT)
measuring quality of class expression (heuristics)
instance checks
query results
query
return quality
test expression
generation of class expressions using a refinement operator

Fig. 27: Outline of the general learning approach in CELOE: One part of the algorithm is the generation of promising class expressions taking the available background knowledge into account. Another part is a heuristic measure of how close an expression is to being a solution of the learning problem. Figure adapted from [55].

Base Learning Algorithm Figure 27 gives a brief overview of the *CELOE* algorithm, which follows the common "generate and test" approach in ILP. This means that learning is seen as a search process and several class expressions are generated and tested against a background knowledge base. Each of those class expressions is evaluated using a heuristic, which is described in the next section. A challenging part of a learning algorithm is to decide which expressions to test. In particular, such a decision should take the computed heuristic values and the structure of the background knowledge into account. For CELOE, we use the approach described in [86,87] as base, where this problem has already been analysed, implemented, and evaluated in depth. It is based on the idea of *refinement operators*:

Definition 7 (refinement operator). *A* quasi-ordering *is a reflexive and transitive relation. In a quasi-ordered space* $(S, \preceq)$ *a* downward (upward) refinement operator ρ *is a mapping from* S *to* 2^S*, such that for any* $C \in S$ *we have that* $C' \in \rho(C)$ *implies* $C' \preceq C$ $(C \preceq C')$*.* C' *is called a* specialisation (generalisation) *of* C*.*

Refinement operators can be used for searching in the space of expressions. As ordering we can use subsumption. (Note that the subsumption relation $\sqsubseteq$ is a quasi-ordering.) If an expression C subsumes an expression D ($D \sqsubseteq C$), then C will cover all examples which are covered by D. This makes subsumption a suitable order for searching in expressions as it allows to prune parts of the search space without losing possible solutions.

The approach we used is a top-down algorithm based on refinement operators as illustrated in Figure 28 (more detailed schemata can be found in the slides[30] of the ontology learning lecture of Reasoning Web 2010 [80]). This means that the first class expression, which will be tested is the most general expression ($\top$), which is then mapped

[30] `http://reasoningweb.org/2010/teaching-material/lehmann.pdf`

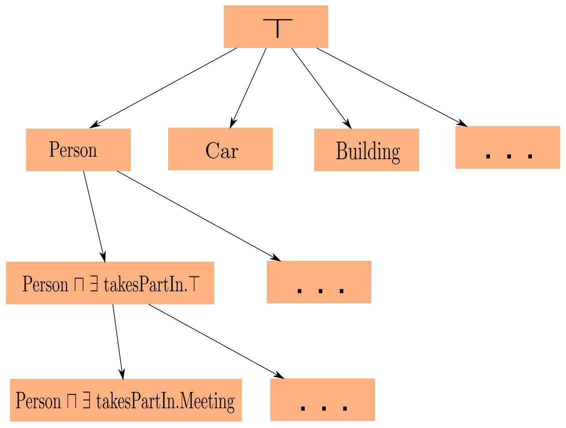


Fig. 28: Illustration of a search tree in a top down refinement approach.

to a set of more specific expressions by means of a downward refinement operator. Naturally, the refinement operator can be applied to the obtained expressions again, thereby spanning a *search tree*. The search tree can be pruned when an expression does not cover sufficiently many instances of the class A we want to describe. One example for a path in a search tree spanned up by a downward refinement operator is the following ($\rightsquigarrow$ denotes a refinement step):

$$\top \rightsquigarrow \texttt{Person} \rightsquigarrow \texttt{Person} \sqcap \texttt{takesPartinIn}.\top \\ \rightsquigarrow \texttt{Person} \sqcap \texttt{takesPartIn.Meeting}$$

The heart of such a learning strategy is to define a suitable refinement operator and an appropriate search heuristics for deciding which nodes in the search tree should be expanded. The refinement operator in the considered algorithm is defined in [87]. It is based on earlier work in [86] which in turn is built on the theoretical foundations of [85]. It has been shown to be the best achievable operator with respect to a set of properties (not further described here), which are used to assess the performance of refinement operators. The learning algorithm supports conjunction, disjunction, negation, existential and universal quantifiers, cardinality restrictions, hasValue restrictions as well as boolean and double datatypes.

6.3 Finding a Suitable Heuristic

A heuristic measures how well a given class expression fits a learning problem and is used to guide the search in a learning process. To define a suitable heuristic, we first need to address the question of how to measure the accuracy of a class expression. We introduce several heuristics, which can be used for *CELOE* and later evaluate them.

We cannot simply use supervised learning from examples directly, since we do not have positive and negative examples available. We can try to tackle this problem by using the existing instances of the class as positive examples and the remaining instances as negative examples. This is illustrated in Figure 29, where $\mathcal{K}$ stands for the knowledge base and A for the class to describe. We can then measure accuracy as the number of correctly classified examples divided by the number of all examples. This can be computed as follows for a class expression C and is known as *predictive accuracy* in Machine Learning:

$$predacc(C) = 1 - \frac{|R(A) \setminus R(C)| + |R(C) \setminus R(A)|}{n} \quad n = |Ind(\mathcal{K})|$$

Here, $Ind(\mathcal{K})$ stands for the set of individuals occurring in the knowledge base. $R(A) \setminus R(C)$ are the false negatives whereas $R(C) \setminus R(A)$ are false positives. n is the number of all examples, which is equal to the number of individuals in the knowledge base in this case. Apart from learning definitions, we also want to be able to learn super class axioms ($A \sqsubseteq C$). Naturally, in this scenario $R(C)$ should be a superset of $R(A)$. However, we still do want $R(C)$ to be as small as possible, otherwise $\top$ would always be a solution. To reflect this in our accuracy computation, we penalise false negatives more than false positives by a factor of t ($t > 1$) and map the result to the interval $[0, 1]$:

$$predacc(C, t) = 1 - 2 \cdot \frac{t \cdot |R(A) \setminus R(C)| + |R(C) \setminus R(A)|}{(t + 1) \cdot n} \quad n = |Ind(\mathcal{K})|$$

While being straightforward, the outlined approach of casting class learning into a standard learning problem with positive and negative examples has the disadvantage that the number of negative examples will usually be much higher than the number of positive examples. As shown in Table 4, this may lead to overly optimistic estimates. More importantly, this accuracy measure has the drawback of having a dependency on the number of instances in the knowledge base.

Therefore, we investigated further heuristics, which overcome this problem, in particular by transferring common heuristics from information retrieval to the class learning problem:

1. *F-Measure:* F_β-Measure is based on *precision* and *recall* weighted by β. They can be computed for the class learning problem without having negative examples. Instead, we perform a retrieval for the expression C, which we want to evaluate. We can then define precision as the percentage of instances of C, which are also instances of A and recall as percentage of instances of A, which are also instances of C. This is visualised in Figure 29. F-Measure is defined as harmonic mean of precision and recall. For learning super classes, we use F_3 measure by default, which gives recall a higher weight than precision.
2. *A-Measure:* We denote the arithmetic mean of precision and recall as A-Measure. Super class learning is achieved by assigning a higher weight to recall. Using the arithmetic mean of precision and recall is uncommon in Machine Learning, since it results in too optimistic estimates. However, we found that it is useful in super class learning, where F_n is often too pessimistic even for higher n.
3. *Generalised F-Measure:* Generalised F-Measure has been published in [36] and extends the idea of F-measure by taking the three valued nature of classification in OWL/DLs into account: An individual can either belong to a class, the negation of a class or none of both cases can be proven. This differs from common binary

classification tasks and, therefore, appropriate measures have been introduced (see [36] for details). Adaption for super class learning can be done in a similar fashion as for F-Measure itself.

4. *Jaccard Distance:* Since $R(A)$ and $R(C)$ are sets, we can use the well-known Jaccard coefficient to measure the similarity between both sets.

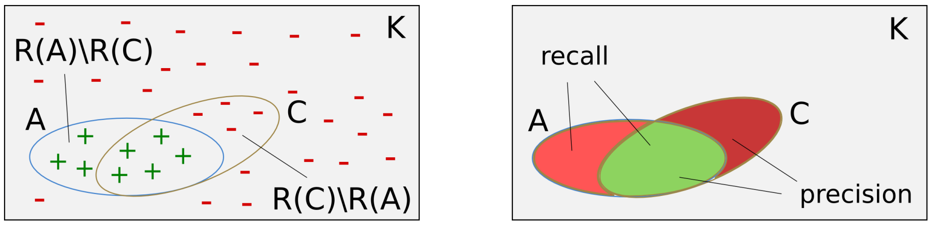


Fig. 29: Visualisation of different accuracy measurement approaches. $\mathcal{K}$ is the knowledge base, A the class to describe and C a class expression to be tested. Left side: Standard supervised approach based on using positive (instances of A) and negative (remaining instances) examples. Here, the accuracy of C depends on the number of individuals in the knowledge base. Right side: Evaluation based on two criteria: recall (*Which fraction of* $R(A)$ *is in* $R(C)$*?*) and precision (*Which fraction of* $R(C)$ *is in* $R(A)$*?*).

We argue that those four measures are more appropriate than predictive accuracy when applying standard learning algorithms to the ontology engineering use case. Table 4 provides some example calculations, which allow the reader to compare the different heuristics.

Efficient Heuristic Computation Several optimisations for computing the heuristics are described in [81]. In particular, adapted approximate reasoning and stochastic approximations are discussed. Those improvements have shown to lead to order of magnitude gains in efficiency for many ontologies. We refrain from describing those methods in this chapter.

The Protégé Plugin After implementing and testing the described learning algorithm, we integrated it into *Protégé* and *OntoWiki*. Together with the Protégé developers, we extended the Protégé 4 plugin mechanism to be able to seamlessly integrate the DL-Learner plugin as an additional method to create class expressions. This means that the knowledge engineer can use the algorithm exactly where it is needed without any additional configuration steps. The plugin has also become part of the official Protégé 4 repository, i.e. it can be directly installed from within Protégé.

A screenshot of the plugin is shown in Figure 30. To use the plugin, the knowledge engineer is only required to press a button, which then starts a new thread in the background. This thread executes the learning algorithm. The used algorithm is an *anytime algorithm*, i.e. at each point in time we can always see the currently best suggestions.

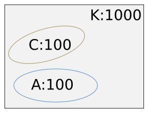
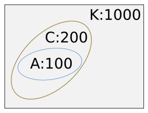
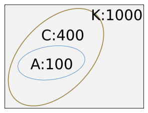
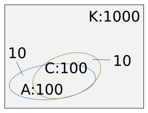
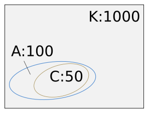
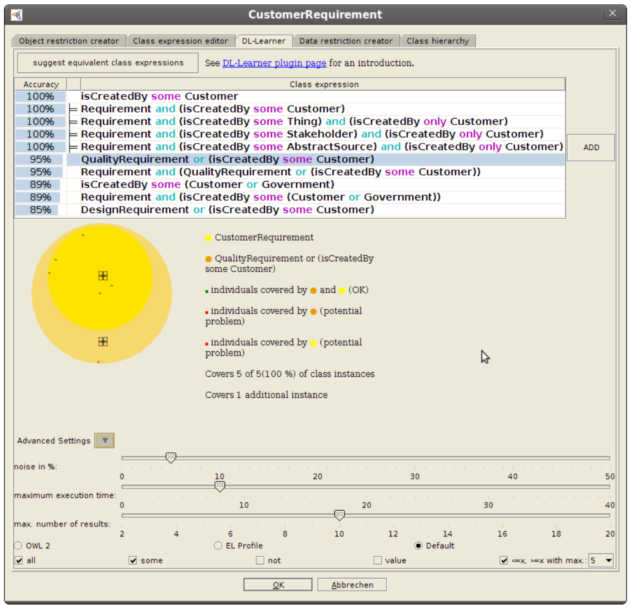
illustration	pred. acc.		F-Measure		A-Measure		Jaccard
	eq	sc	eq	sc	eq	sc	
	80%	67%	0%	0%	0%	0%	0%
	90%	92%	67%	73%	75%	88%	50%
	70%	75%	40%	48%	63%	82%	25%
	98%	97%	90%	90%	90%	90%	82%
	95%	88%	67%	61%	75%	63%	50%

Table 4: Example accuracies for selected cases (eq = equivalence class axiom, sc = super class axiom). The images on the left represent an imaginary knowledge base $\mathcal{K}$ with 1000 individuals, where we want to describe the class A by using expression C. It is apparent that using predictive accuracy leads to impractical accuracies, e.g. in the first row C cannot possibly be a good description of A, but we still get 80% accuracy, since all the negative examples outside of A and C are correctly classified.

The GUI updates the suggestion list each second until the maximum runtime – 10 seconds by default – is reached. This means that the perceived runtime, i.e. the time after which only minor updates occur in the suggestion list, is often only one or two seconds for small ontologies. For each suggestion, the plugin displays its accuracy.

When clicking on a suggestion, it is visualized by displaying two circles: One stands for the instances of the class to describe and another circle for the instances of the suggested class expression. Ideally, both circles overlap completely, but in practice this will often not be the case. Clicking on the plus symbol in each circle shows its list of individuals. Those individuals are also presented as points in the circles and moving the mouse over such a point shows information about the respective individual. Red points show potential problems detected by the plugin. Please note that we use closed world reasoning to detect those problems. For instance, in our initial example, a capital which is not related via the property `isCapitalOf` to an instance of `Country` is marked red. If there is not only a potential problem, but adding the expression would render the ontology inconsistent, the suggestion is marked red and a warning message is displayed.


CustomerRequirement
Object restriction creator
Class expression editor
DL-Learner
Data restriction creator
Class hierarchy
suggest equivalent class expressions
See DL-Learner plugin page for an introduction.
ADD
● CustomerRequirement
● QualityRequirement or (isCreatedBy some Customer)
▪ individuals covered by ● and ● (OK)
▪ individuals covered by ● (potential problem)
▪ individuals covered by ● (potential problem)
Advanced Settings
OK
Abbrechen

Accuracy		Class expression
100%		isCreatedBy some Customer
100%	⊨	Requirement and (isCreatedBy some Customer)
100%	⊨	Requirement and (isCreatedBy some Thing) and (isCreatedBy only Customer)
100%	⊨	Requirement and (isCreatedBy some Stakeholder) and (isCreatedBy only Customer)
100%	⊨	Requirement and (isCreatedBy some AbstractSource) and (isCreatedBy only Customer)
95%		QualityRequirement or (isCreatedBy some Customer)
95%		Requirement and (QualityRequirement or (isCreatedBy some Customer))
89%		isCreatedBy some (Customer or Government)
89%		Requirement and (isCreatedBy some (Customer or Government))
85%		DesignRequirement or (isCreatedBy some Customer)

Covers 5 of 5(100 %) of class instances

Covers 1 additional instance

noise in %: 0 10 20 30 40 50

maximum execution time: 0 10 20 30 40

max. number of results: 2 4 6 8 10 12 14 16 18 20

○ OWL 2 ○ EL Profile [✓ ● Default]

[✓] all [✓] some [ ] not [ ] value [✓] <=x, >=x with max.: 5

Fig. 30: A screenshot of the DL-Learner Protégé plugin. It is integrated as additional tab to create class expressions in Protégé. The user is only required to press the "suggest equivalent class expressions" button and within a few seconds they will be displayed ordered by accuracy. If desired, the knowledge engineer can visualize the instances of the expression to detect potential problems. At the bottom, optional expert configuration settings can be adopted.

Accepting such a suggestion can still be a good choice, because the problem often lies elsewhere in the knowledge base, but was not obvious before, since the ontology was not sufficiently expressive for reasoners to detect it. This is illustrated by a screencast available from the plugin homepage,[31] where the ontology becomes inconsistent after adding the axiom, and the real source of the problem is fixed afterwards. Being able to make such suggestions can be seen as a strength of the plugin.

The plugin allows the knowledge engineer to change expert settings. Those settings include the maximum suggestion search time, the number of results returned and settings related to the desired target language, e.g. the knowledge engineer can choose to stay within the OWL 2 EL profile or enable/disable certain class expression constructors. The learning algorithm is designed to be able to handle noisy data and the visualisation of the suggestions will reveal false class assignments so that they can be fixed afterwards.

The OntoWiki Plugin Analogous to Protégé, we created a similar plugin for OntoWiki (cf. section 4). OntoWiki is a lightweight ontology editor, which allows distributed and collaborative editing of knowledge bases. It focuses on wiki-like, simple and intuitive authoring of semantic content, e.g. through inline editing of RDF content, and provides different views on instance data.

Recently, a fine-grained plugin mechanism and extensions architecture was added to OntoWiki. The DL-Learner plugin is technically realised by implementing an OntoWiki component, which contains the core functionality, and a module, which implements the

[31] http://dl-learner.org/wiki/ProtegePlugin

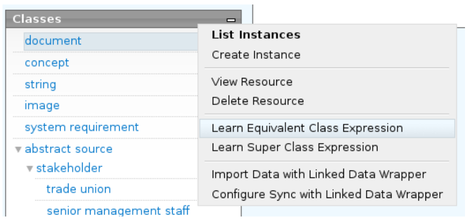


Fig. 31: The DL-Learner plugin can be invoked from the context menu of a class in OntoWiki.

UI embedding. The DL-Learner plugin can be invoked from several places in OntoWiki, for instance through the context menu of classes as shown in Figure 31.

The plugin accesses DL-Learner functionality through its WSDL-based web service interface. Jar files containing all necessary libraries are provided by the plugin. If a user invokes the plugin, it scans whether the web service is online at its default address. If not, it is started automatically.

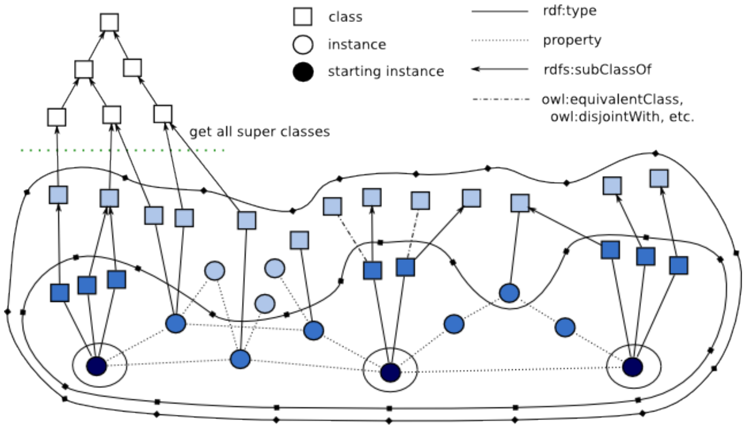
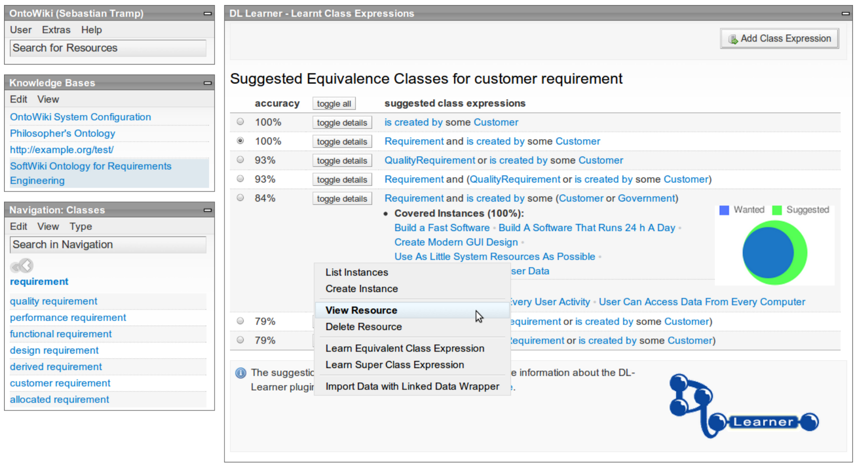


Fig. 32: Extraction with three starting instances. The circles represent different recursion depths. The circles around the starting instances signify recursion depth 0. The larger inner circle represents the fragment with recursion depth 1 and the largest outer circle with recursion depth 2. Figure taken from [55].

A major technical difference compared to the Protégé plugin is that the knowledge base is accessed via SPARQL, since OntoWiki is a SPARQL-based web application. In Protégé, the current state of the knowledge base is stored in memory in a Java object. As a result, we cannot easily apply a reasoner on an OntoWiki knowledge base. To overcome this problem, we use the DL-Learner fragment selection mechanism described in [55,56,30]. Starting from a set of instances, the mechanism extracts a relevant fragment from the underlying knowledge base up to some specified recursion depth. Figure 32 provides an overview of the fragment selection process. The fragment has the property that learning results on it are similar to those on the complete knowledge base. For a detailed description we refer the reader to the full article.

The fragment selection is only performed for medium to large-sized knowledge bases. Small knowledge bases are retrieved completely and loaded into the reasoner. While the fragment selection can cause a delay of several seconds before the learning algorithm starts, it also offers flexibility and scalability. For instance, we can learn class expressions in large knowledge bases such as DBpedia in OntoWiki.[32]



Suggested Equivalence Classes for customer requirement

accuracy	toggle all	suggested class expressions
100%	toggle details	is created by some Customer
100%	toggle details	Requirement and is created by some Customer
93%	toggle details	QualityRequirement or is created by some Customer
93%	toggle details	Requirement and (QualityRequirement or is created by some Customer)
84%	toggle details	Requirement and is created by some (Customer or Government)
		• Covered Instances (100%): Build a Fast Software · Build A Software That Runs 24 h A Day · Create Modern GUI Design · Use As Little System Resources As Possible · ser Data · very User Activity · User Can Access Data From Every Computer
79%		equirement or is created by some Customer)
79%		equirement or is created by some Customer)

The suggestic e information about the DL-Learner plugir .

Fig. 33: Screenshot of the result table of the DL-Learner plugin in OntoWiki.

Figure 33 shows a screenshot of the OntoWiki plugin applied to the SWORE [129] ontology. Suggestions for learning the class "customer requirement" are shown in Manchester OWL Syntax. Similar to the Protégé plugin, the user is presented a table of suggestions along with their accuracy value. Additional details about the instances of "customer requirement", covered by a suggested class expressions and additionally contained instances can be viewed via a toggle button. The modular design of OntoWiki allows rich user interaction: Each resource, e.g. a class, property, or individual, can be viewed and subsequently modified directly from the result table as shown for "design requirement" in the screenshot. For instance, a knowledge engineer could decide to import additional information available as Linked Data and run the CELOE algorithm again to see whether different suggestions are provided with additional background knowledge.

Evaluation To evaluate the suggestions made by our learning algorithm, we tested it on a variety of real-world ontologies of different sizes and domains. Please note that we

[32] OntoWiki is undergoing an extensive development, aiming to support handling such large knowledge bases. A release supporting this is expected for the first half of 2012.

intentionally do not perform an evaluation of the machine learning technique as such on existing benchmarks, since we build on the base algorithm already evaluated in detail in [87]. It was shown that this algorithm is superior to other supervised learning algorithms for OWL and at least competitive with the state of the art in ILP. Instead, we focus on its use within the ontology engineering scenario. The goals of the evaluation are to 1. determine the influence of reasoning and heuristics on suggestions, 2. to evaluate whether the method is sufficiently efficient to work on large real-world ontologies.

To perform the evaluation, we wrote a dedicated plugin for the Protégé ontology editor. This allows the evaluators to browse the ontology while deciding whether the suggestions made are reasonable. The plugin works as follows: First, all classes with at least 5 inferred instances are determined. For each such class, we run CELOE with different settings to generate suggestions for definitions. Specifically, we tested two reasoners and five different heuristics. The two reasoners are standard Pellet and Pellet combined with approximate reasoning (not described in detail here). The five heuristics are those described in Section 6.3. For each configuration of CELOE, we generate at most 10 suggestions exceeding a heuristic threshold of 90%. Overall, this means that there can be at most 2 * 5 * 10 = 100 suggestions per class – usually less, because different settings of CELOE will still result in similar suggestions. This list is shuffled and presented to the evaluators. For each suggestion, the evaluators can choose between 6 options (see Table 6):

1 the suggestion improves the ontology (improvement)
2 the suggestion is no improvement and should not be included (not acceptable) and
3 adding the suggestion would be a modelling error (error)

In the case of existing definitions for class A, we removed them prior to learning. In this case, the evaluator could choose between three further options:

4 the learned definition is equal to the previous one and both are good (equal +)
5 the learned definition is equal to the previous one and both are bad (equal -) and
6 the learned definition is inferior to the previous one (inferior).

We used the default settings of CELOE, e.g. a maximum execution time of 10 seconds for the algorithm. The knowledge engineers were five experienced members of our research group, who made themselves familiar with the domain of the test ontologies. Each researcher worked independently and had to make 998 decisions for 92 classes between one of the options. The time required to make those decisions was approximately 40 working hours per researcher. The raw agreement value of all evaluators is 0.535 (see e.g. [4] for details) with 4 out of 5 evaluators in strong pairwise agreement (90%). The evaluation machine was a notebook with a 2 GHz CPU and 3 GB RAM.

[33] http://www.mindswap.org/ontologies/SC.owl
[34] http://www.sbcny.org/datasets/adhesome.owl
[35] http://i2geo.net/ontologies/current/GeoSkills.owl
[36] http://www.co-ode.org/ontologies/eukariotic/2005/06/01/eukariotic.owl
[37] http://acl.icnet.uk/%7Emw/MDM0.73.owl
[38] http://reliant.teknowledge.com/DAML/Economy.owl
[39] http://www.ecs.soton.ac.uk/~aoj04r/resist.owl
[40] http://www.fadyart.com/Finance.owl
[41] http://sweet.jpl.nasa.gov/1.1/earthrealm.owl

Ontology	#logical axioms	#classes	#object properties	#data properties	#individuals	DL expressivity
SC Ontology[33]	20081	28	8	5	3542	$\mathcal{AL}(\mathcal{D})$
Adhesome[34]	12043	40	33	37	2032	$\mathcal{ALCHN}(\mathcal{D})$
GeoSkills[35]	14966	613	23	21	2620	$\mathcal{ALCHOIN}(\mathcal{D})$
Eukariotic[36]	38	11	1	0	11	$\mathcal{ALCON}$
Breast Cancer[37]	878	196	22	3	113	$\mathcal{ALCROF}(\mathcal{D})$
Economy[38]	1625	339	45	8	482	$\mathcal{ALCH}(\mathcal{D})$
Resist[39]	239	349	134	38	75	$\mathcal{ALUF}(\mathcal{D})$
Finance[40]	16014	323	247	74	2466	$\mathcal{ALCROIQ}(\mathcal{D})$
Earthrealm[41]	931	2364	215	36	171	$\mathcal{ALCHO}(\mathcal{D})$

Table 5: Statistics about test ontologies

reasoner/heuristic	improvement	equal quality (+)	equal quality (-)	inferior	not acceptable	error	missed improvements in %	selected position on suggestion list (incl. std. deviation)	avg. accuracy of selected suggestion in %
Pellet/F-Measure	16.70	0.44	0.66	0.00	64.66	17.54	14.95	2.82 ± 2.93	96.91
Pellet/Gen. F-Measure	15.24	0.44	0.66	0.11	66.60	16.95	16.30	2.78 ± 3.01	92.76
Pellet/A-Measure	16.70	0.44	0.66	0.00	64.66	17.54	14.95	2.84 ± 2.93	98.59
Pellet/pred. acc.	16.59	0.44	0.66	0.00	64.83	17.48	15.22	2.69 ± 2.82	98.05
Pellet/Jaccard	16.81	0.44	0.66	0.00	64.66	17.43	14.67	2.80 ± 2.91	95.26
Pellet FIC/F-Measure	36.30	0.55	0.55	0.11	52.62	9.87	1.90	2.25 ± 2.74	95.01
Pellet FIC/Gen. F-M.	33.41	0.44	0.66	0.00	53.41	12.09	7.07	1.77 ± 2.69	89.42
Pellet FIC/A-Measure	36.19	0.55	0.55	0.00	52.84	9.87	1.63	2.21 ± 2.71	98.65
Pellet FIC/pred. acc.	32.99	0.55	0.55	0.11	55.58	10.22	4.35	2.17 ± 2.55	98.92
Pellet FIC/Jaccard	36.30	0.55	0.55	0.11	52.62	9.87	1.90	2.25 ± 2.74	94.07

Table 6: Options chosen by evaluators aggregated by class. FIC stands for the fast instance checker, which is an approximate reasoning procedure.

Table 6 shows the evaluation results. All ontologies were taken from the Protégé OWL[42] and TONES[43] repositories. We randomly selected 5 ontologies comprising instance data from these two repositories, specifically the Earthrealm, Finance, Resist, Economy and Breast Cancer ontologies (see Table 5).

The results in Table 6 show which options were selected by the evaluators. It clearly indicates that the usage of approximate reasoning is sensible. The results are, however, more difficult to interpret with regard to the different employed heuristics. Using predictive accuracy did not yield good results and, surprisingly, generalised F-Measure also had a lower percentage of cases where option 1 was selected. The other three heuristics generated very similar results. One reason is that those heuristics are all based on precision and recall, but in addition the low quality of some of the randomly selected test ontologies posed a problem. In cases of too many very severe modelling errors, e.g. conjunctions and disjunctions mixed up in an ontology or inappropriate domain and range restrictions, the quality of suggestions decreases for each of the heuristics. This is the main reason why the results for the different heuristics are very close. Particularly, generalised F-Measure can show its strengths mainly for properly designed ontologies. For instance, column 2 of Table 6 shows that it missed 7% of possible improvements. This means that for 7% of all classes, one of the other four heuristics was able to find an appropriate definition, which was not suggested when employing generalised F-Measure. The last column in this table shows that the average value of generalised F-Measure is quite low. As explained previously, it distinguishes between cases when an individual is instance of the observed class expression, its negation, or none of both. In many cases, the reasoner could not detect that an individual is instance of the negation of a class expression, because of the absence of disjointness axioms and negation in the knowledge base, which explains the low average values of generalised F-Measure. Column 4 of Table 6 shows that many selected expressions are amongst the top 5 (out of 10) in the suggestion list, i.e. providing 10 suggestions appears to be a reasonable choice.

In general, the improvement rate is only at about 35% according to Table 6 whereas it usually exceeded 50% in preliminary experiments with other real-world ontologies with fewer or less severe modelling errors. Since CELOE is based on OWL reasoning, it is clear that schema modelling errors will have an impact on the quality of suggestions. As a consequence, we believe that the CELOE algorithm should be combined with ontology debugging techniques. We have obtained first positive results in this direction and plan to pursue it in future work. However, the evaluation also showed that CELOE does still work in ontologies, which probably were never verified by an OWL reasoner.

Summary We presented the CELOE learning method specifically designed for extending OWL ontologies. Five heuristics were implemented and analysed in conjunction with CELOE along with several performance improvements. A method for approximating heuristic values has been introduced, which is useful beyond the ontology engineering scenario to solve the challenge of dealing with a large number of examples in ILP [159]. Furthermore, we biased the algorithm towards short solutions and implemented optimisations to increase readability of the suggestions made. The resulting

[42] `http://protegewiki.stanford.edu/index.php/Protege_Ontology_Library`
[43] `http://owl.cs.manchester.ac.uk/repository/`

algorithm was implemented in the open source DL-Learner framework. We argue that CELOE is the first ILP based algorithm, which turns the idea of learning class expressions for extending ontologies into practice. CELOE is integrated into two plugins for the ontology editors Protégé and OntoWiki and can be invoked using just a few mouse clicks.

7 Linked Data Quality

Linked Open Data (LOD) has provided, over the past several years, an unprecedented volume of structured data currently amount to 50 billion facts, represented as RDF triples. Although publishing large amounts of data on the Web is certainly a step in the right direction, the published data is only as useful as its quality. On the Data Web we have very varying quality of information covering various domains since data is merged together from different autonomous evolving data sources on the Web. For example, data extracted from semi-structured or even unstructured sources, such as DBpedia, often contains inconsistencies as well as mis-represented and incomplete information. Despite data quality in LOD being an essential concept, the autonomy and openness of the information providers makes the web vulnerable to missing, inaccurate, incomplete, inconsistent or outdated information.

Data quality is commonly conceived as *fitness for use* [71,158] for a certain application or use case. However, even datasets with quality problems might be useful for certain applications, as long as the quality is in the required range. For example, in the case of DBpedia the data quality is perfectly sufficient for enriching Web search with facts or suggestions about common sense information, such as entertainment topics. In such a scenario, DBpedia can be used to show related movies and personal information, when a user searches for an actor. In this case, it is rather neglectable, when in relatively few cases, a related movie or some personal fact is missing. For developing a medical application, on the other hand, the quality of DBpedia is probably completely insufficient. It should be noted that even the traditional, document-oriented Web has content of varying quality and is still perceived to be extremely useful by most people.

Consequently, one of the key challenges is to determine the quality of datasets published on the Web and making this quality information explicitly available. Assuring data quality is particularly a challenge in LOD as it involves a set of autonomously evolving data sources. Other than on the document Web, where information quality can be only indirectly (e.g. via page rank) or vaguely defined, there are much more concrete and measurable data quality metrics available for structured information such as accuracy of facts, completeness, adequacy of semantic representation or degree of understandability.

In this chapter, we first define the basic concepts of data quality, then report the formal definitions of a set of 26 different dimensions along with their respective metrics identified in [166].

Thereafter, we describe a set of currently available tools specially designed to assess the quality of Linked Data.

7.1 Data Quality Concepts

In this section, we introduce the basic concepts of data quality to help the readers understand these terminologies in their consequent usage.

Data Quality. The term *data quality* is commonly conceived as a multi-dimensional construct with a popular definition as the "fitness for use" [71]. In case of the Semantic Web, there are varying concepts of data quality such as the semantic metadata on the one hand and the notion of link quality on the other. There are several characteristics of data quality that should be considered i.e. the completeness, accuracy, consistency and validity on the one hand and the representational consistency, conciseness as well as the timeliness, understandability, availability and verifiability on the other hand.

Data Quality Problems. A set of issues that can affect the potentiality of the applications that use the data are termed as data quality problems. The problems may vary from the incompleteness of data, inconsistency in representation, invalid syntax or inaccuracy.

Data Quality Dimensions and Metrics. Data quality assessment involves the measurement of quality *dimensions* (or *criteria*) that are relevant to the user. A data quality assessment *metric* (or *measure*) is a procedure for measuring an information quality dimension [24]. The metrics are basically heuristics designed to fit a specific assessment situation [88]. Since the dimensions are rather abstract concepts, the assessment metrics rely on quality *indicators* that can be used for the assessment of the quality of a data source w.r.t the criteria [46].

Data Quality Assessment Method. A data quality assessment methodology is the process of evaluating if a piece of data meets the information consumers need for a specific use case [24]. The process involves measuring the quality dimensions that are relevant to the user and comparing the assessment results with the users quality requirements.

7.2 Linked Data quality dimensions

In [166], a core set of 26 different data quality dimensions were reported that can be applied to assess the quality of Linked Data. These dimensions are divided into the following groups:

- Contextual dimensions
- Trust dimensions
- Intrinsic dimensions
- Accessibility dimensions
- Representational dimensions
- Dataset dynamicity

Figure 34 shows the classification of the dimensions into these 6 different groups as well as the relations between them.

Use case scenario. Since data quality is described as "fitness to use", we introduce a specific use case that will allow us to illustrate the importance of each dimension with the help of an example. Our use case is about an intelligent flight search engine, which relies on acquiring (aggregating) data from several datasets. It obtains information about airports and airlines from an airline dataset (e.g., OurAirports[44], OpenFlights[45]). Information about the location of countries, cities and particular addresses is obtained

[44] `http://thedatahub.org/dataset/ourairports`
[45] `http://thedatahub.org/dataset/open-flights`

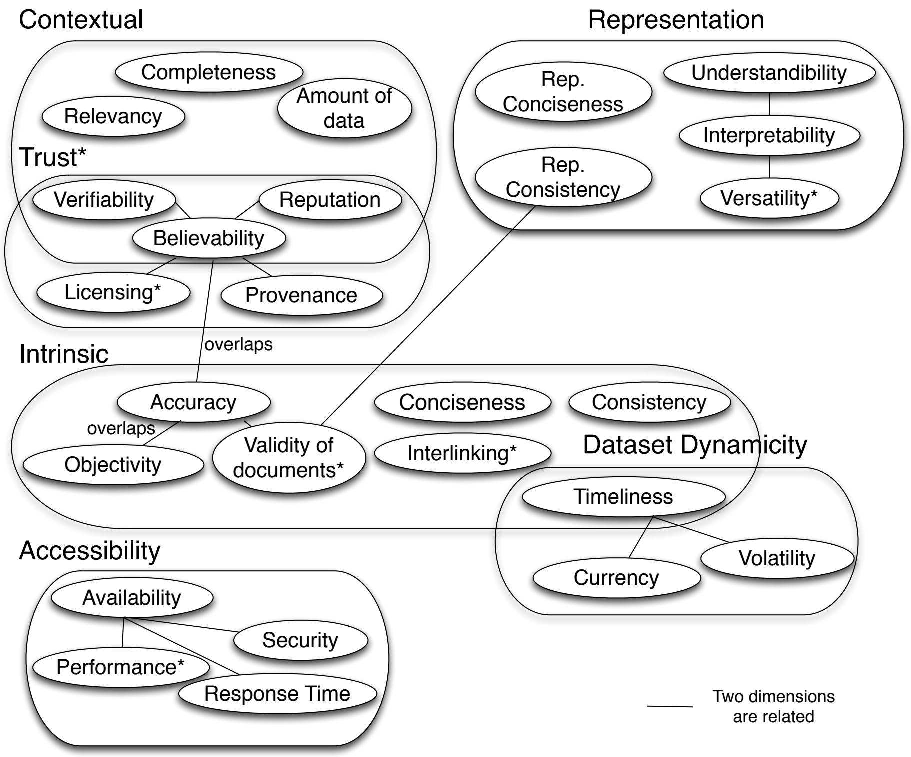


Fig. 34: Linked data quality dimensions and the relations between them [Source: [166].]

from a spatial dataset (e.g., LinkedGeoData[46]). Additionally, aggregators in RDF pull all the information related to airlines from different data sources (e.g., Expedia[47], Tripadvisor[48], Skyscanner[49] etc.) that allows a user to query the integrated dataset for a flight from any start and end destination for any time period. We will use this scenario throughout this section as an example of how data quality influences fitness to use.

Contextual dimensions. Contextual dimensions are those that highly depend on the context of the task at hand as well as on the subjective preferences of the data consumer. There are three dimensions *completeness*, *amount-of-data* and *relevancy* that are part of this group.

Definition 8 (Completeness). *Completeness refers to the degree to which all required information is present in a particular dataset. In general, completeness is the extent to*

[46] `linkedgeodata.org`
[47] `http://www.expedia.com/`
[48] `http://www.tripadvisor.com/`
[49] `http://www.skyscanner.de/`

which data is of sufficient depth, breadth and scope for the task at hand. In terms of Linked Data, we classify completeness as follows:

- *Schema completeness, the degree to which the classes and properties of an ontology are represented, thus can be called "ontology completeness",*
- *Property completeness, measure of the missing values for a specific property,*
- *Population completeness, the percentage of all real-world objects of a particular type that are represented in the datasets and*
- *Interlinking completeness, which has to be considered especially in Linked Data and refers to the degree to which instances in the dataset are interlinked.*

Metrics. Completeness can be measured by detecting the number of classes, properties, values and interlinks that are present in the dataset by comparing it to the original dataset (or gold standard dataset). It should be noted that in this case, users should assume a closed-world-assumption where a gold standard dataset is available and can be used to compare against.

Example. In the use case, the flight search engine should contain complete information so that it includes all offers for flights (population completeness). For example, a user residing in Germany wants to visit her friend in America. Since the user is a student, low price is of high importance. But, looking for flights individually on the airlines websites shows her flights with very expensive fares. However, using our flight search engine she finds all offers, even the less expensive ones and is also able to compare fares from different airlines and choose the most affordable one. The more complete the information for flights is, including cheap flights, the more visitors the site attracts. Moreover, sufficient interlinks between the datasets will allow her to query the integrated dataset so as to find an optimal route from the start to the end destination (interlinking completeness) in cases when there is no direct flight.

Definition 9 (Amount-of-data). *Amount-of-data refers to the quantity and volume of data that is appropriate for a particular task.*

Metrics. The amount-of-data can be measured in terms of bytes (most coarse-grained), triples, instances, and/or links present in the dataset. This amount should represent an appropriate volume of data for a particular task, that is, appropriate scope and level of detail.

Example. In the use case, the flight search engine acquires enough amount of data so as to cover all, even small, airports. In addition, the data also covers alternative means of transportation. This helps to provide the user with better travel plans, which includes smaller cities (airports). For example, when a user wants to travel from Connecticut to Santa Barbara, she might not find direct or indirect flights by searching individual flights websites. But, using our example search engine, she is suggested convenient flight connections between the two destinations, because it contains a large amount of data so as to cover all the airports. She is also offered convenient combinations of planes, trains and buses. The provision of such information also necessitates the presence of a large amount of internal as well as externals links between the datasets so as to provide a fine grained search for flights between specific places.

Definition 10 (Relevancy). *Relevancy refers to the provision of information which is in accordance with the task at hand and important to the users' query.*

Metrics. Relevancy is highly context dependent and can be measured by using meta-information attributes for assessing whether the content is relevant for a particular task. Additionally, retrieval of relevant documents can be performed using a combination of hyperlink analysis and information retrieval methods.

Example. When a user is looking for flights between any two cities, only relevant information i.e. start and end times, duration and cost per person should be provided. If a lot of irrelevant data is included in the spatial data, e.g. post offices, trees etc. (present in LinkedGeoData), query performance can decrease. The user may thus get lost in the silos of information and may not be able to browse through it efficiently to get only what she requires.

Trust dimensions. Trust dimensions are those that focus on the trustworthiness of the dataset. There are five dimensions that are part of this group, namely, *provenance*, *verifiability*, *believability*, *reputation* and *licensing*.

Definition 11 (Provenance). *Provenance refers to the contextual metadata that focuses on how to represent, manage and use information about the origin of the source. Provenance helps to describe entities to enable trust, assess authenticity and allow reproducibility.*

Metrics. Provenance can be measured by analyzing the metadata associated with the source. This provenance information can in turn be used to assess the trustworthiness, reliability and credibility of a data source, an entity, a publishers or even individual RDF statements. There exists an inter-dependancy between the data provider and the data itself. On the one hand, data is likely to be accepted as true if it is provided by a trustworthy provider. On the other hand, the data provider is trustworthy if it provides true data. Thus, both can be checked to measure the trustworthiness.

Example. The example flight search engine constitutes information from several airline providers. In order to verify the reliability of these different airline data providers, provenance information from the aggregators can be analyzed and re-used so as enable users of the flight search engine to trust the authenticity of the data provided to them.

Definition 12 (Verifiability). *Verifiability refers to the degree by which a data consumer can assess the correctness of a dataset and as a consequence its trustworthiness.*

Metrics. Verifiability can be measured either by an unbiased third party, if the dataset itself points to the source or by the presence of a digital signature.

Example. In the use case, if we assume that the flight search engine crawls information from arbitrary airline websites, which publish flight information according to a standard vocabulary, there is a risk for receiving incorrect information from malicious websites. For instance, such a website publishes cheap flights just to attract a large number of visitors. In that case, the use of digital signatures for published RDF data allows to restrict crawling only to verified datasets.

Definition 13 (Reputation). *Reputation is a judgement made by a user to determine the integrity of a source. It is mainly associated with a data publisher, a person, organisation, group of people or community of practice rather than being a characteristic of a dataset. The data publisher should be identifiable for a certain (part of a) dataset.*

Metrics. Reputation is usually a score, for example, a real value between 0 (low) and 1 (high). There are different possibilities to determine reputation and can be classified into manual or (semi-)automated approaches. The manual approach is via a survey in a community or by questioning other members who can help to determine the reputation of a source or by the person who published a dataset. The (semi-)automated approach can be performed by the use of external links or page ranks.

Example. The provision of information on the reputation of data sources allows conflict resolution. For instance, several data sources report conflicting prices (or times) for a particular flight number. In that case, the search engine can decide to trust only the source with a higher reputation.

Definition 14 (Believability). *Believability is defined as the degree to which the information is accepted to be correct, true, real and credible.*

Metrics. Believability is measured by checking whether the contributor is contained in a list of trusted providers. In Linked Data, believability can be subjectively measured by analyzing the provenance information of the dataset.

Example. In the flight search engine use case, if the flight information is provided by trusted and well-known flights companies such as Lufthansa, British Airways, etc. then the user believes the information provided by their websites. She does not need to verify their credibility since these are well-known international flight companies. On the other hand, if the user retrieves information about an airline previously unknown, she can decide whether to believe this information by checking whether the airline is well-known or if it is contained in a list of trusted providers. Moreover, she will need to check the source website from which this information was obtained.

Definition 15 (Licensing). *Licensing is defined as a granting of the permission for a consumer to re-use a dataset under defined conditions.*

Metrics. Licensing can be checked by the indication of machine and human readable information associated with the dataset clearly indicating the permissions of data re-use.

Example. Since the example flight search engine aggregates data from several data sources, a clear indication of the license allows the search engine to re-use the data from the airlines websites. For example, the LinkedGeoData dataset is licensed under the Open Database License[50], which allows others to copy, distribute and use the data and produce work from the data allowing modifications and transformations. Due to the presence of this specific license, the flight search engine is able to re-use this dataset to pull geo-spatial information and feed it to the search engine.

[50] `http://opendatacommons.org/licenses/odbl/`

Intrinsic dimensions. Intrinsic dimensions are those that are independent of the user's context. These dimensions focus on whether information correctly represents the real world and whether information is logically consistent in itself. There are five dimensions that are part of this group, namely, *accuracy*, *objectivity*, *validity-of-documents*, *interlinking*, *consistency* and *conciseness*.

Definition 16 (Accuracy). *Accuracy can be defined as the extent to which data is correct, that is, the degree to which it correctly represents the real world facts and is also free of error. In particular, we associate accuracy mainly to semantic accuracy which relates to the correctness of a value to the actual real world value, that is, accuracy of the meaning.*

Metrics. Accuracy can be measured by checking the correctness of the data in a data source. That is, the detection of outliers or identification of semantically incorrect values through the violation of functional dependency rules. Accuracy is one of the dimensions, which is affected by assuming a closed or open world. When assuming an open world, it is more challenging to assess accuracy, since more logical constraints need to be specified for inferring logical contradictions.

Example. In the use case, let us suppose that a user is looking for flights between Paris and New York. Instead of returning flights starting from Paris, France, the search returns flights between Paris in Texas and New York. This kind of semantic inaccuracy in terms of labelling as well as classification can lead to erroneous results.

Definition 17 (Objectivity). *Objectivity is defined as the degree to which the interpretation and usage of data is unbiased, unprejudiced and impartial. This dimension highly depends on the type of information and therefore is classified as a subjective dimension.*

Metrics. Objectivity can not be measured qualitatively but indirectly by checking the authenticity of the source responsible for the information, whether the dataset is neutral or the publisher has a personal influence on the data provided. Additionally, it can be measured by checking whether independent sources can confirm a single fact.

Example. In the example flight search engine, consider the reviews available for each airline regarding the safety, comfort and prices. It may happen that an airline belonging to a particular alliance is ranked higher than others when in reality it is not so. This could be an indication of a bias where the review is falsified due to the providers preference or intentions. This kind of bias or partiality affects the user as she might be provided with incorrect information from expensive flights or from malicious websites.

Definition 18 (Validity-of-documents). *Validity-of-documents refers to the valid usage of the underlying vocabularies and the valid syntax of the documents (syntactic accuracy).*

Metrics. A syntax validator can be employed to assess the validity of a document, i.e. its syntactic correctness. The syntactic accuracy of entities can be measured by detecting the erroneous or inaccurate annotations, classifications or representations. An RDF validator can be used to parse the RDF document and ensure that it is syntactically valid, that is, to check whether the document is in accordance with the RDF specification.

Example. Let us assume that the user is looking for flights between two specific locations, for instance Paris (France) and New York (United States) using the example flight search engine. However, the user is returned with no results. A possible reason for this is that one of the data sources incorrectly uses the property `geo:lon` to specify the longitude of "Paris" instead of using `geo:long`. This causes a query to retrieve no data when querying for flights starting near to a particular location.

Definition 19 (Interlinking). *Interlinking refers to the degree to which entities that represent the same concept are linked to each other.*

Metrics. Interlinking can be measured by using network measures that calculate the interlinking degree, cluster coefficient, sameAs chains, centrality and description richness through sameAs links.

Example. In the example flight search engine, the instance of the country `"United States"` in the airline dataset should be interlinked with the instance `"America"` in the spatial dataset. This interlinking can help when a user queries for a flight as the search engine can display the correct route from the start destination to the end destination by correctly combining information for the same country from both the datasets. Since names of various entities can have different URIs in different datasets, their interlinking can help in disambiguation.

Definition 20 (Consistency). *Consistency means that a knowledge base is free of (logical/formal) contradictions with respect to particular knowledge representation and inference mechanisms.*

Metrics. On the Linked Data Web, semantic knowledge representation techniques are employed, which come with certain inference and reasoning strategies for revealing implicit knowledge, which then might render a contradiction. Consistency is relative to a particular logic (set of inference rules) for identifying contradictions. A consequence of our definition of consistency is that a dataset can be consistent wrt. the RDF inference rules, but inconsistent when taking the OWL2-QL reasoning profile into account. For assessing consistency, we can employ an inference engine or a reasoner, which supports the respective expressivity of the underlying knowledge representation formalism. Additionally, we can detect functional dependency violations such as domain/range violations. In practice, RDF-Schema inference and reasoning with regard to the different OWL profiles can be used to measure consistency in a dataset. For domain specific applications, consistency rules can be defined, for example, according to the SWRL [63] or RIF standards [72] and processed using a rule engine.

Example. Let us assume a user is looking for flights between Paris and New York on the 21st of December, 2012. Her query returns the following results:

```
Flight   From    To          Arrival   Departure
A123     Paris   NewYork     14:50     22:35
A123     Paris   Singapore   14:50     22:35
```

The results show that the flight number `A123` has two different destinations at the same date and same time of arrival and departure, which is inconsistent with the ontology definition that one flight can only have one destination at a specific time and date. This contradiction arises due to inconsistency in data representation, which can be detected by using inference and reasoning.

Definition 21 (Conciseness). *Conciseness refers to the redundancy of entities, be it at the schema or the data level. Thus, conciseness can be classified into:*

- *intensional conciseness (schema level) which refers to the redundant attributes and*
- *extensional conciseness (data level) which refers to the redundant objects.*

Metrics. As conciseness is classified in two categories, it can be measured by as the ratio between the number of unique attributes (properties) or unique objects (instances) compared to the overall number of attributes or objects respectively present in a dataset.

Example. In the example flight search engine, since data is fused from different datasets, an example of intensional conciseness would be a particular flight, say `A123`, being represented by two different identifiers in different datasets, such as `http://airlines.org/A123` and `http://flights.org/A123`. This redundancy can ideally be solved by fusing the two and keeping only one unique identifier. On the other hand, an example of extensional conciseness is when both these different identifiers of the same flight have the same information associated with them in both the datasets, thus duplicating the information.

Accessibility dimensions. The dimensions belonging to this category involve aspects related to the way data can be accessed and retrieved. There are four dimensions part of this group, which are *availability*, *performance*, *security* and *response-time*.

Definition 22 (Availability). *Availability of a dataset is the extent to which information is present, obtainable and ready for use.*

Metrics. Availability of a dataset can be measured in terms of accessibility of the server, SPARQL endpoints or RDF dumps and also by the dereferencability of the URIs.

Example. Let us consider the case in which the user looks up a flight in the example flight search engine. However, instead of retrieving the results, she is presented with an error response code such as `4xx client error`. This is an indication that a requested resource is unavailable. In particular, when the returned error code is `404 Not Found code`, she may assume that either there is no information present at that specified URI or the information is unavailable. Naturally, an apparently unreliable system is less likely to be used, in which case the user may not book flights after encountering such issues.

Definition 23 (Performance). *Performance refers to the efficiency of a system that binds to a large dataset, that is, the more performant a data source the more efficiently a system can process data.*

Metrics. Performance is measured based on the scalability of the data source, that is a query should be answered in a reasonable amount of time. Also, detection of the usage of prolix RDF features or usage of slash-URIs can help determine the performance of a dataset. Additional metrics are low latency and high throughput of the services provided for the dataset.

Example. In the use case example, the target performance may depend on the number of users, i.e. it may be required to be able to server 100 simultaneous users. Our flight search engine will not be scalable if the time required to answer to all queries is similar to the time required when querying the individual datasets. In that case, satisfying performance needs requires caching mechanisms.

Definition 24 (Security). *Security can be defined as the extent to which access to data can be restricted and hence protected against its illegal alteration and misuse. It refers to the degree to which information is passed securely from users to the information source and back.*

Metrics. Security can be measured based on whether the data has a proprietor or requires web security techniques (e.g. SSL or SSH) for users to access, acquire or re-use the data. The importance of security depends on whether the data needs to be protected and whether there is a cost of data becoming unintentionally available. For open data the protection aspect of security can be often neglected but the non-repudiation of the data is still an important issue. Digital signatures based on private-public key infrastructures can be employed to guarantee the authenticity of the data.

Example. In the example scenario, let us consider a user who wants to book a flight from a city `A` to a city `B`. The search engine should ensure a secure environment to the user during the payment transaction since her personal data is highly sensitive. If there is enough identifiable public information of the user, then she can be potentially targeted by private businesses, insurance companies etc. which she is unlikely to want. Thus, the use of SSL can be used to keep the information safe.

Definition 25 (Response-time). *Response-time measures the delay, usually in seconds, between submission of a query by the user and reception of the complete response from the dataset.*

Metrics. Response-time can be assessed by measuring the delay between submission of a request by the user and reception of the response from the dataset.

Example. A user is looking for a flight which includes multiple destinations using the example flight search engine. She wants to fly from `Milan` to `Boston`, `Boston` to `New York` and `New York` to `Milan`. In spite of the complexity of the query the search engine should respond quickly with all the flight details (including connections) for all the destinations.

Representational dimensions. Representational dimensions capture aspects related to the design of the data such as the *representational-conciseness*, *representational-consistency*, *understandability*, *versatility* as well as the *interpretability* of the data.

Definition 26 (Representational-conciseness). *Representational-conciseness refers to the representation of the data which is compact and well formatted on the one hand but also clear and complete on the other hand.*

Metrics. Representational-conciseness is measured by qualitatively verifying whether the RDF model that is used to represent the data is concise enough in order to be self-descriptive and unambiguous.

Example. A user, after booking her flight, is interested in additional information about the destination airport such as its location. Our flight search engine should provide only that information related to the location rather than returning a chain of other properties.

Definition 27 (Representational-consistency). *Representational-consistency is the degree to which the format and structure of the information conform to previously returned information. Since Linked Data involves aggregation of data from multiple sources, we extend this definition to not only imply compatibility with previous data but also with data from other sources.*

Metrics. Representational-consistency can be assessed by detecting whether the dataset re-uses existing vocabularies or terms from existing established vocabularies to represent its entities.

Example. In the use case, consider different airlines companies using different notation for representing their data, e.g. some use RDF data and some others use turtle. In order to avoid interoperability issue, we provide data based on the Linked Data principle which is designed to support heterogeneous description models, which is necessary to handle different format of data. The exchange of information in different formats will not be a big deal in our search engine since strong links are created between datasets.

Definition 28 (Understandability). *Understandability refers to the ease with which data can be comprehended, without ambiguity, and used by a human consumer. Thus, this dimension can also be referred to as the comprehensibility of the information where the data should be of sufficient clarity in order to be used.*

Metrics. Understandability can be measured by detecting whether human-readable labels for classes, properties and entities are provided. Provision of the metadata of a dataset can also contribute towards assessing its understandability. The dataset should also clearly provide exemplary URIs and SPARQL queries along with the vocabularies used so that can users can understand how it can be used.

Example. Let us assume that the example flight search engine allows a user to enter a start and destination address. In that case, strings entered by the user need to be matched to entities in the spatial dataset4, probably via string similarity. Understandable labels for cities, places etc. improve search performance in that case. For instance, when a user looks for a flight to U.S (label), then the search engine should return the flights to the United States or America.

Definition 29 (Interpretability). *Interpretability refers to technical aspects of the data, that is whether information is represented using an appropriate notation and whether it conforms to the technical ability of the consumer.*

Metrics. Interpretability can be measured by the use of globally unique identifiers for objects and terms or by the use of appropriate language, symbols, units and clear definitions.

Example. Consider the example flight search engine wherein a user that is looking for a flight from `Milan` to `Boston`. Data related to Boston in the integrated data, for the required flight, contains the following entities:

– `http://rdf.freebase.com/ns/m.049jnng`
– `http://rdf.freebase.com/ns/m.043j22x`
– Boston Logan Airport

For the first two items no human-readable label is available, therefore the URI is displayed, which does not represent anything meaningful to the user besides the information that Freebase contains information about Boston Logan Airport. The third, however, contains a human-readable label, which the user can easily interpret.

Definition 30 (Versatility). *Versatility mainly refers to the alternative representations of data and its subsequent handling. Additionally, versatility also corresponds to the provision of alternative access methods for a dataset.*

Metrics. Versatility can be measured by the availability of the dataset in different serialisation formats, different languages as well as different access methods.

Example. Consider a user from a non-English speaking country who wants to use the example flight search engine. In order to cater to the needs of such users, our flight search engine should be available in different languages so that any user has the capability to understand it.

Dataset dynamicity. An important aspect of data is its update or change over time. The main dimensions related to the dynamicity of a dataset proposed in the literature are *currency*, *volatility*, and *timeliness*.

Definition 31 (Currency). *Currency refers to the speed with which the information (state) is updated after the real-world information changes.*

Metrics. The measurement of currency relies on two components: (i) delivery time (the time when the data was last modified) and (ii) the current time, both possibly present in the data models.

Example. Consider a user who is looking for the price of a flight from Milan to Boston using the example search engine and she receives the updates via email. The currency of the price information is measured with respect to the last update of the price information. The email service sends the email with a delay of about 1 hour with respect to the time interval in which the information is determined to hold. In this way, if the currency value exceeds the time interval of the validity of the information, the result is said to be not current. If we suppose validity of the price information (the frequency of change of the price information) to be 2 hours, the price list that is not changed within this time is considered to be out-dated. To determine whether the information is out-dated or not we need to have temporal metadata available and represented by one of the data models proposed in the literature [133].

Definition 32 (Volatility). *Volatility can be defined as the length of time during which the data remains valid.*

Metrics. Volatility can be measured by two components: (i) the expiry time (the time when the data becomes invalid) and (ii) the input time (the time when the data was first published on the Web). Both these metrics are combined together to measure the distance between the expiry time and the input time of the published data.

Example. Let us consider the aforementioned use case where a user wants to book a flight from Milan to Boston and she is interested in the price information. The price is considered as volatile information as it changes frequently, for instance it is estimated that the flight price changes each minute (data remains valid for one minute). In order to have an updated price list, the price should be updated within the time interval pre-defined by the system (one minute). In case the user observes that the value is not re-calculated by the system within the last few minutes, the values are considered to be out-dated. Notice that volatility is estimated based on changes that are observed related to a specific data value.

Definition 33 (Timeliness). *Timeliness refers to the time point at which the data is actually used. This can be interpreted as whether the information is available in time to be useful.*

Metrics. Timeliness is measured by combining the two dimensions: currency and volatility. Additionally timeliness states the recency and frequency of data validation and does not include outdated data.

Example. A user wants to book a flight from Milan to Boston and the example flight search engine will return a list of different flight connections. She picks one of them and follows all the steps to book her flight. The data contained in the airline dataset shows a flight company that is available according to the user requirements. In terms of time-related quality dimension, the information related to the flight is recorded and reported to the user every two minutes which fulfils the requirement decided by our search engine that corresponds to the volatility of a flight information. Although the flight values are updated on time, the information received to the user about the flight's availability is not on time. In other words, the user did not perceive that the availability of the flight was depleted because the change was provided to the system a moment after her search.

7.3 Data quality assessment frameworks

There are several efforts in developing data quality assessment frameworks (methodologies) in order to assess the data quality of LOD. These efforts are either semi-automated [46], automated [51] or manual [24,99,151]. Moreover, there is a lot of research performed extensively to assess the quality of LOD [60,61] and report commonly occurring problems that plague the existing datasets. In this section, we describe the various existing data quality assessment tools in terms of their usability, level of user interaction and applicability in terms of data quality assessment, also discussing their pros and cons.

Semi-automated. Flemming's data quality assessment tool [46], a semi-automated framework, is a simple user interface[51]. The tool presents an evaluation system to the user where she needs to only enter the SPARQL endpoint and a few example resources from a dataset. After specifying dataset details (endpoint, graphs, example URIs), the user is given an option for assigning weights to each of the pre-defined data quality metrics. Two options are available for assigning weights: (a) assigning a weight of 1

[51] Available in German only at: `http://linkeddata.informatik.hu-berlin.de/LDSrcAss/datenquelle.php`

to all the metrics or (b) choosing the pre-defined exemplary weight of the metrics defined for a semantic data source. In the next step, the user is asked to answer a series of questions regarding the datasets, which are important indicators of the data quality for Linked Datasets and those which cannot be quantified. These include, for example, questions about the use of stable URIs, the number of obsolete classes and properties, and whether the dataset provides a mailing list. Next, the user is presented with a list of dimensions and metrics, for each of which weights can be specified again. Each metric is provided with two input fields: one showing the assigned weights and another with the calculated value.

In the end, the user is presented with a score ranging from 0 to 100, where 100 represents the best quality, representing the final data quality score. Additionally, the rating of each dimension and the total weight (based on 11 dimensions) is calculated using the user input from the previous steps.

On one hand, the tool is easy to use with the form-based questions and adequate explanation for each step. Also, the assigning of the weights for each metric and the calculation of the score is straightforward and easy to adjust for each of the metrics. However, this tool has a few drawbacks: (1) the user needs to have adequate knowledge about the dataset in order to correctly assign weights for each of the metrics; (2) it does not drill down to the root cause of the data quality problem and (3) some of the main quality dimensions are missing from the analysis such as accuracy, completeness, provenance, consistency, conciseness and relevancy as some could not be quantified and were not perceived to be true quality indicators.

Automated. The *LINK-QA framework* [51] takes a set of resources, SPARQL endpoints and/or dereferencable resources and a set of triples as input to automatically perform quality assessment and generates an HTML report. However, using this tool, the user cannot choose the dataset that she is interested in assessing.

Manual. The *WIQA* [24] and *Sieve* [99] frameworks also assess the quality of datasets but require a considerable amount of user involvement and are therefore considered manual tools. For instance, WIQA provides the user a wide range of policies to filter information and a set of quality criteria to assess the quality of the information.

Sieve, on the other hand, assists not only in the assessment of the quality of datasets but also in their fusion. It aims to use the data integration task as a means to increase completeness, conciseness and consistency in any chosen dataset. Sieve is a component of the *Linked Data Integration Framework* (LDIF)[52] used first to assess the quality between two or more data sources and second to fuse (integrate) the data from the data sources based on their quality assessment.

In order to use this tool, a user needs to be conversant with programming. The input of Sieve is an LDIF provenance metadata graph generated from a data source. Based on this information the user needs to set the configuration property in an XML file known as `integration properties`. The quality assessment procedure relies on the measurement of metrics chosen by the user where each metric applies a scoring function having a value from 0 to 1.

[52] `http://ldif.wbsg.de/`

Sieve implements only a few scoring functions such as `TimeCloseness`, `Preference`, `SetMembership`, `Threshold` and `Interval Membership` which are calculated based on the metadata provided as input along with the original data source. The configuration file is in XML format which should be modified based on the use case, as shown in Listing 1.1. The output scores are then used to fuse the data sources by applying one of the fusion functions, which are: `Filter`, `Average`, `Max`, `Min`, `First`, `KeepSingleValue ByQualityScore`, `Last`, `Random`, `PickMostFrequent`.

Listing 1.1: A configuration of Sieve: a data quality assessment and data fusion tool.

```
<Sieve>
 <QualityAssessment>
  <AssessmentMetric id="sieve:recency">
  <ScoringFunction class="TimeCloseness">
    <Param name="timeSpan" value="7"/>
    <Input path="?GRAPH/provenance:lasUpdated"/>
  </ScoringFunction>
  </AssessmentMetric>
  <AssessmentMetric id="sieve:reputation">
  <ScoringFunction class="ScoredList">
    <Param name="priority" value="http://pt.wikipedia.org http://en.wikipedia.org"/>
    <Input path="?GRAPH/provenance:lasUpdated"/>
  </ScoringFunction>
  </AssessmentMetric>
</Sieve>
```

In addition, users should specify the `DataSource` folder, the `homepage` element that refers to the data source from which the entities are going to be fused. Second, the XML file of the `ImportJobs` that downloads the data to the server should also be modified. In particular, the user should set up the `dumpLocation` element as the location of the dump file.

Although the tool is very useful overall, there are some drawbacks that decreases its usability: (1) the tool is not mainly used to assess data quality for a source, but instead to perform data fusion (integration) based on quality assessment. Therefore, the quality assessment can be considered as an accessory that leverages the process of data fusion through evaluation of few quality indicators; (2) it does not provide a user interface, ultimately limiting its usage to end-users with programming skills; (3) its usage is limited to domains providing provenance metadata associated with the data source.

LODGRefine[53] [151], a LOD-enabled version of Google Refine, is an open-source tool for refining messy data. Although this tool is not focused on data quality assessment per se, it is powerful in performing preliminary cleaning or refining of raw data. Using this tool, one is able to import several different file types of data (CSV, Excel, XML, RDF/XML, N-Triples or even JSON) and then perform cleaning action via a browser-based interface. By using a diverse set of filters and facets on individual columns, LODGRefine can help a user to semi-automate the cleaning of her data.

For example, this tool can help to detect duplicates, discover patterns (e.g. alternative forms of an abbreviation), spot inconsistencies (e.g. trailing white spaces) or find and replace blank cells. Additionally, this tool allows users to reconcile data, that is to connect a dataset to existing vocabularies such that it gives meaning to the values. Rec-

[53] `http://code.zemanta.com/sparkica/`

onciliations to *Freebase*[54] helps mapping ambiguous textual values to precisely identified Freebase entities. Reconciling using *Sindice* or based on standard SPARQL or SPARQL with full-text search is also possible[55] using this tool. Moreover, it is also possible to extend the reconciled data with DBpedia as well as export the data as RDF, which adds to the uniformity and usability of the dataset.

These feature thus assists in assessing as well as improving the data quality of a dataset. Moreover, by providing external links, the interlinking of the dataset is considerably improved. LODGRefine is easy to download and install as well as to upload and perform basic cleansing steps on raw data. The features of reconciliation, extending the data with DBpedia, transforming and exporting the data as RDF are added advantages. However, this tool has a few drawbacks: (1) the user is not able to perform detailed high level data quality analysis utilizing the various quality dimensions using this tool; (2) performing cleansing over a large dataset is time consuming as the tool follows a column data model and thus the user must perform transformations per column.

[54] `http://www.freebase.com/`
[55] `http://refine.deri.ie/reconciliationDocs`

8 Outlook and Future Challenges

Although the different approaches for aspects of the Linked Data life-cycle as presented in this chapter are already working together, more effort must be done to further integrate them in ways that they mutually fertilize themselves. The discovery of new links or the authoring of new resource descriptions, for example, should automatically trigger the enrichment of the linked knowledge bases. The enrichment in turn can trigger the application of inconsistency detection and repair techniques. This leads to recognizing data quality problems and their consequent assessment and improvement. The browsing and exploration paths followed by end-users can be taken into account for machine learning techniques to refine the knowledge bases etc. Ultimately, when the different aspects of Linked Data management are fully integrated we envision the Web of Data becoming a washing machine for knowledge. A progress in one particular aspect will automatically trigger improvements in many other ones as well. In the following we outline some research challenges and promising research directions regarding some of the Linked Data management aspects.

Extraction. One promising research direction with regard to the extraction from unstructured sources is the development of standardized, LOD enabled integration interfaces between existing NLP tools. An open question is whether and how efficient bidirectional synchronization between extraction source and target knowledge base can be established. With regard to the extraction from structured sources (e.g. relational, XML) we need a declarative syntax and semantics for data model transformations. Some orthogonal challenges include the use of LOD as background knowledge and the representation and tracing of provenance information.

Authoring. Current Semantic Wikis still suffer from a lack of scalability. Hence, an important research and development target are large-scale Semantic Wikis, which include functionality for access control and provenance. In order to further flexibilize and simplify the authoring an adaptive choreography of editing widgets based on underlying data structures is needed. Also, the joint authoring of unstructured and structured sources (i.e. HTML/RDFa authoring) and better support for the integrated semantic annotation of other modalities such as images, audio, video is of paramount importance.

Natural Language Queries. One of the future challenges for Linked Data is to create user interfaces, which are able to hide the complexity of the underlying systems. A possible path towards this goal is question answering, e.g. converting natural language queries to SPARQL [149,84]. In order to allow users to interact with such systems, there is ongoing work on converting the created SPARQL queries back to natural language [112] and employ feedback mechanisms [83,58]. Ultimately, a goal is to provide users enhanced functionality without the need to adapt to different kinds of interface.

Automatic Management of Resources for Linking. With the growth of the Cloud and of the datasets that need to be interlinked, the use of parallel hardware has been studied over the last few years [59,113]. The comparative study of parallel hardware for link discovery yet shows surprising results and suggests that the use of massively parallel yet local hardware can lead to tremendous runtime improvements. Still, when result sets go beyond sizes of 10^{10}, the higher amount of resources available on remote devices in the Cloud is still to be used. Devising automatic solutions for selecting the right hardware to run a linking task is one of the most interesting research areas pertaining to the efficient execution of link specifications. In addition, developing reduction-ratio optimal algorithms for spaces other than Minkowski spaces promises to ensure the best possible use of available hardware. Finally, devising more efficient means to combine single algorithms is the third open area of research in this domain. The challenges faces with regard to learning link specifications are also manifold and include devising approaches that can efficiently detected most informative positive and negative examples as well even running in a fully unsupervised manner on properties that are not one-to-one relations.

Linked Data Visualization The potential of the vast amount of Linked Data on the Web is enormous but in most cases it is very difficult *and* cumbersome for users to visualize, explore and use this data, especially for lay-users [**?**] without experience with Semantic Web technologies. Visualizations are useful for obtaining an overview of the datasets, their main types, properties and the relationships between them. Compared to prior information visualization strategies, we have a unique opportunity on the Data Web. The unified RDF data model being prevalent on the Data Web enables us to bind data to visualizations in an *unforeseen* and *dynamic* way. An information visualization technique requires certain data structures to be present. When we can derive and generate these data structures automatically from reused vocabularies or semantic representations, we are able to realize a largely automatic visualization workflow. Ultimately, various visualizations techniques can develop an ecosystem of data extractions and visualizations, which can be bound together in a dynamic and unforeseen way. This will enable users to explore datasets even if the publisher of the data does not provide any exploration or visualization means. Yet, most existing work related to visualizing RDF is focused on concrete domains and concrete datatypes.

Acknowledgments

We would like to thank our colleagues from the AKSW research group in Leipzig as well as the LOD2 and GeoKnow project consortia, without whom writing this chapter would not have been possible. In particular, we would like to thank Christian Bizer and Tom Heath, whose Chaper 2 of the book 'Linked Data – Evolving the Web into a Global Data Space' [53] served as a blueprint for Section 2; Sebastian Hellmann, Claus Stadler, Jörg Unbehauen for their contributions to Section 3, Sebastian Tramp, Michael Martin, Norman Heino, Phillip Frischmuth and Thomas Riechert for their contributions to the development of OntoWiki as described in Section 4. This work was supported by a grant from the European Union's 7th Framework Programme provided for the projects LOD2 (GA no. 257943), GeoKnow (GA no. 318159) and the Eureka project SCMS.

References

1. Resource description framework (RDF): Concepts and abstract syntax. Technical report, W3C, 2 2004.
2. Ben Adida, Mark Birbeck, Shane McCarron, and Steven Pemberton. RDFa in XHTML: Syntax and processing – a collection of attributes and processing rules for extending XHTML to support RDF. W3C Recommendation, Oct 2008. `http://www.w3.org/TR/rdfa-syntax/`.
3. Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *In ACM DL*, pages 85–94, 2000.
4. Alan Agresti. *An Introduction to Categorical Data Analysis 2nd edition*. Wiley-Interscience, 1997.
5. R. Amsler. Research towards the development of a lexical knowledge base for natural language processing. *SIGIR Forum*, 23:1–2, 1989.
6. Sören Auer, Chris Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *ISWC 2007*, volume 4825 of *LNCS*, pages 722–735. Springer.
7. Sören Auer, Lorenz Bühmann, Jens Lehmann, Michael Hausenblas, Sebastian Tramp, Bert van Nuffelen, Pablo Mendes, Christian Dirschl, Robert Isele, Hugh Williams, and Orri Erling. Managing the life-cycle of linked data with the lod2 stack. In *Proceedings of International Semantic Web Conference (ISWC 2012)*, 2012. 22
8. Sören Auer, Sebastian Dietzold, Jens Lehmann, Sebastian Hellmann, and David Aumueller. Triplify: Light-weight linked data publication from relational databases. In Juan Quemada, Gonzalo León, Yoëlle S. Maarek, and Wolfgang Nejdl, editors, *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*, pages 621–630. ACM, 2009.
9. Sören Auer, Sebastian Dietzold, and Thomas Riechert. OntoWiki - A Tool for Social, Semantic Collaboration. In Isabel F. Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Michael Uschold, and Lora Aroyo, editors, *The Semantic Web - ISWC 2006, 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5-9, 2006, Proceedings*, volume 4273 of *Lecture Notes in Computer Science*, pages 736–749, Berlin / Heidelberg, 2006. Springer.
10. Sören Auer and Heinrich Herre. A versioning and evolution framework for rdf knowledge bases. In *Proceedings of at Sixth International Andrei Ershov Memorial Conference - Perspectives of System Informatics (PSI'06), 27-30 June, Novosibirsk, Akademgorodok, Russia*, volume 4378, June 2006.

11. Sören Auer and Jens Lehmann. Making the web a data washing machine - creating knowledge out of interlinked data. *Semantic Web Journal*, 2010.
12. Sören Auer, Jens Lehmann, and Sebastian Hellmann. LinkedGeoData - adding a spatial dimension to the web of data. In *Proc. of 8th International Semantic Web Conference (ISWC)*, 2009.
13. Sören Auer, Jens Lehmann, and Axel-Cyrille Ngonga Ngomo. Introduction to linked data and its lifecycle on the web. In *Reasoning Web*, pages 1–75, 2011.
14. David Aumüller. Semantic Authoring and Retrieval within a Wiki (WikSAR). In Demo Session at the Second European Semantic Web Conference (ESWC2005), May 2005. Available at http://wiksar.sf.net, 2005.
15. Franz Baader, Bernhard Ganter, Ulrike Sattler, and Baris Sertkaya. Completing description logic knowledge bases using formal concept analysis. In *IJCAI 2007*. AAAI Press, 2007.
16. Franz Baader, Baris Sertkaya, and Anni-Yasmin Turhan. Computing the least common subsumer w.r.t. a background terminology. *J. Applied Logic*, 5(3):392–420, 2007.
17. Liviu Badea and Shan-Hwei Nienhuys-Cheng. A refinement operator for description logics. In *ILP 2000*, volume 1866 of *LNAI*, pages 40–59. Springer, 2000.
18. Rohan Baxter, Peter Christen, and Tim Churches. A comparison of fast blocking methods for record linkage. In *KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, 2003.
19. David Ben-David, Tamar Domany, and Abigail Tarem. Enterprise data classification using semantic web technologies. In *ISWC2010*, 2010.
20. Tim Berners-Lee. Notation 3, 1998. See `http://www.w3.org/DesignIssues/Notation3.html`.
21. Tim Berners-Lee, Roy Thomas Fielding, and Larry Masinter. Uniform resource identifiers (URI): Generic syntax. Internet RFC 2396, August 1998.
22. Ravish Bhagdev, Sam Chapman, Fabio Ciravegna, Vitaveska Lanfranchi, and Daniela Petrelli. Hybrid search: Effectively combining keywords and semantic searches. In *ESWC 2008*, pages 554–568, 2008.
23. Mikhail Bilenko, Beena Kamath, and Raymond J. Mooney. Adaptive blocking: Learning to scale up record linkage. In *ICDM '06*, pages 87–96. IEEE, 2006.
24. Christian Bizer and Richard Cyganiak. Quality-driven information filtering using the wiqa policy framework. *Web Semantics*, 7(1):1 – 10, Jan 2009.
25. Jens Bleiholder and Felix Naumann. Data fusion. *ACM Comput. Surv.*, 41(1):1–41, 2008.
26. Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. In *Readings in Machine Learning*, pages 201–204. Morgan Kaufmann, 1990.
27. Dan Brickley and Ramanatgan V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. W3C recommendation, W3C, February 2004. `http://www.w3.org/TR/2004/REC-rdf-schema-20040210/`.
28. Sergey Brin. Extracting patterns and relations from the world wide web. In *WebDB*, pages 172–183, London, UK, 1999. Springer-Verlag.
29. Lorenz Bühmann and Jens Lehmann. Universal OWL axiom enrichment for large knowledge bases. In *Proceedings of EKAW 2012*, 2012.
30. Didier Cherix, Sebastian Hellmann, and Jens Lehmann. Improving the performance of a sparql component for semantic web applications. In *JIST*, 2012.
31. Namyoun Choi, Il-Yeol Song, and Hyoil Han. A survey on ontology mapping. *SIGMOD Record*, 35(3):34–41, 2006.
32. Sam Coates-Stephens. The analysis and acquisition of proper names for the understanding of free text. *Computers and the Humanities*, 26:441–456, 1992. 10.1007/BF00136985.
33. William W. Cohen, Alexander Borgida, and Haym Hirsh. Computing least common subsumers in description logics. In *AAAI 1992*, pages 754–760, 1992.

34. William W. Cohen and Haym Hirsh. Learning the CLASSIC description logic: Theoretical and experimental results. In *KR 1994*, pages 121–133. Morgan Kaufmann, 1994.
35. James R. Curran and Stephen Clark. Language independent ner using a maximum entropy tagger. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4*, pages 164–167, Morristown, NJ, USA, 2003. Association for Computational Linguistics.
36. Claudia d'Amato, Nicola Fanizzi, and Floriana Esposito. A note on the evaluation of inductive concept classification procedures. In Aldo Gangemi, Johannes Keizer, Valentina Presutti, and Heiko Stoermer, editors, *SWAP 2008*, volume 426 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.
37. Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19:1–16, 2007.
38. Timofey Ermilov, Norman Heino, Sebastian Tramp, and Sören Auer. OntoWiki Mobile — Knowledge Management in your Pocket. In *Proceedings of the ESWC2011*, 2011.
39. Floriana Esposito, Nicola Fanizzi, Luigi Iannone, Ignazio Palmisano, and Giovanni Semeraro. Knowledge-intensive induction of terminologies from metadata. In *ISWC 2004*, pages 441–455. Springer, 2004.
40. Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: an experimental study. *Artif. Intell.*, 165:91–134, June 2005.
41. Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007.
42. Nicola Fanizzi, Claudia d'Amato, and Floriana Esposito. DL-FOIL concept learning in description logics. In *ILP 2008*, volume 5194 of *LNCS*, pages 107–121. Springer, 2008.
43. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – http/1.1 (rfc 2616). Request For Comments, 1999. available at `http://www.ietf.org/rfc/rfc2616.txt`, accessed 7 July 2006.
44. Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Morristown, NJ, USA, 2005. Association for Computational Linguistics.
45. Daniel Fleischhacker, Johanna Völker, and Heiner Stuckenschmidt. Mining rdf data for property axioms. In *OTM Conferences (2)*, pages 718–735, 2012.
46. Annika Flemming. Quality characteristics of linked data publishing datasources. Master's thesis, Humboldt-Universität zu Berlin, 2010.
47. Eibe Frank, Gordon W. Paynter, Ian H. Witten, Carl Gutwin, and Craig G. Nevill-Manning. Domain-specific keyphrase extraction. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, IJCAI '99, pages 668–673, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.
48. Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, 2007.
49. Hugh Glaser, Ian C. Millard, Won-Kyung Sung, Seungwoo Lee, Pyung Kim, and Beom-Jong You. Research on linked data and co-reference resolution. Technical report, University of Southampton, 2009.
50. R. Grishman and R. Yangarber. Nyu: Description of the Proteus/Pet system as used for MUC-7 ST. In *MUC-7*. Morgan Kaufmann, 1998.
51. Christophe Guéret, Paul Groth, Claus Stadler, and Jens Lehmann. Assessing linked data mappings using network measures. In *ESWC*, 2012.
52. Sanda Harabagiu, Cosmin Adrian Bejan, and Paul Morarescu. Shallow semantics for relation extraction. In *IJCAI*, pages 1061–1066, 2005.

53. Tom Heath and Christian Bizer. *Linked Data - Evolving the Web into a Global Data Space*, volume 1 of *Synthesis Lectures on the Semantic Web:Theory and Technology*. Morgan & Claypool, 2011.
54. Norman Heino, Sebastian Dietzold, Michael Martin, and Sören Auer. Developing semantic web applications with the ontowiki framework. In Tassilo Pellegrini, Sören Auer, Klaus Tochtermann, and Sebastian Schaffert, editors, *Networked Knowledge - Networked Media*, volume 221 of *Studies in Computational Intelligence*, pages 61–77. Springer, Berlin / Heidelberg, 2009.
55. Sebastian Hellmann, Jens Lehmann, and Sören Auer. Learning of OWL class descriptions on very large knowledge bases. *Int. J. Semantic Web Inf. Syst.*, 5(2):25–48, 2009.
56. Sebastian Hellmann, Jens Lehmann, and Sören Auer. Learning of owl class expressions on very large knowledge bases and its applications. In Interoperability Semantic Services and Web Applications: Emerging Concepts, editors, *Learning of OWL Class Expressions on Very Large Knowledge Bases and its Applications*, chapter 5, pages 104–130. IGI Global, 2011.
57. Sebastian Hellmann, Jens Lehmann, and Sören Auer. Linked-data aware uri schemes for referencing text fragments. In *EKAW 2012*, Lecture Notes in Computer Science (LNCS) 7603. Springer, 2012.
58. Sebastian Hellmann, Jens Lehmann, Jörg Unbehauen, Claus Stadler, Thanh Nghia Lam, and Markus Strohmaier. Navigation-induced knowledge engineering by example. In *JIST*, 2012.
59. Stanley Hillner and Axel-Cyrille Ngonga Ngomo. Parallelizing limes for large-scale link discovery. In *I'Semantics*, 2011.
60. Aidan Hogan, Andreas Harth, Alexandre Passant, Stefan Decker, and Axel Polleres. Weaving the pedantic web. In *LDOW*, 2010.
61. Aidan Hogan, Jürgen Umbrich, Andreas Harth, Richard Cyganiak, Axel Polleres, and Stefan Decker. An empirical survey of linked data conformance. *Journal of Web Semantics*, 2012.
62. Matthew Horridge and Peter F. Patel-Schneider. Manchester syntax for OWL 1.1. In *OWLED 2008*, 2008.
63. I Horrocks, P.F. Patel-Schneider, H Boley, S Tabet, B Grosof, and M Dean. Swrl: A semantic web rule language combining owl and ruleml. Technical report, W3C, May 2004.
64. HTML 5: A vocabulary and associated APIs for HTML and XHTML. W3C Working Draft, August 2009. `http://www.w3.org/TR/2009/WD-html5-20090825/`.
65. Luigi Iannone and Ignazio Palmisano. An algorithm based on counterfactuals for concept learning in the semantic web. In *IEA/AIE 2005*, pages 370–379, June 2005.
66. Luigi Iannone, Ignazio Palmisano, and Nicola Fanizzi. An algorithm based on counterfactuals for concept learning in the semantic web. *Applied Intelligence*, 26(2):139–159, 2007.
67. Ali Inan, Murat Kantarcioglu, Elisa Bertino, and Monica Scannapieco. A hybrid approach to private record linkage. *ICDE*, 0:496–505, 2008.
68. Robert Isele, Anja Jentzsch, and Christian Bizer. Efficient multidimensional blocking for link discovery without losing recall. In *WebDB*, 2011.
69. Robert Isele, Anja Jentzsch, and Christian Bizer. Active learning of expressive linkage rules for the web of data. In *ICWE*, pages 411–418, 2012.
70. Ian Jacobs and Norman Walsh. Architecture of the world wide web, volume one. World Wide Web Consortium, Recommendation REC-webarch-20041215, December 2004.
71. Joseph Juran. *The Quality Control Handbook*. McGraw-Hill, New York, 1974.
72. Michael Kifer and Harold Boley. Rif overview. Technical report, W3C, June 2010. `http://www.w3.org/TR/2012/NOTE-rif-overview-20121211/`.

73. Su Nam Kim and Min-Yen Kan. Re-examining automatic keyphrase extraction approaches in scientific articles. In *Proceedings of the Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation and Applications*, MWE '09, pages 9–16, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.
74. Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 21–26, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.
75. Hanna Köpcke, Andreas Thor, and Erhard Rahm. Comparative evaluation of entity resolution approaches with fever. *Proc. VLDB Endow.*, 2(2):1574–1577, 2009.
76. Markus Krötzsch, Denny Vrandecic, Max Völkel, Heiko Haller, and Rudi Studer. Semantic wikipedia. *Journal of Web Semantics*, 5:251–261, September 2007.
77. Jens Lehmann. Hybrid learning of ontology classes. In Petra Perner, editor, *Machine Learning and Data Mining in Pattern Recognition, 5th International Conference, MLDM 2007, Leipzig, Germany, July 18-20, 2007, Proceedings*, volume 4571 of *Lecture Notes in Computer Science*, pages 883–898. Springer, 2007.
78. Jens Lehmann. DL-Learner: learning concepts in description logics. *Journal of Machine Learning Research (JMLR)*, 10:2639–2642, 2009.
79. Jens Lehmann. *Learning OWL Class Expressions*. PhD thesis, University of Leipzig, 2010. PhD in Computer Science.
80. Jens Lehmann. Ontology learning. In *Proceedings of Reasoning Web Summer School*, 2010.
81. Jens Lehmann, Sören Auer, Lorenz Bühmann, and Sebastian Tramp. Class expression learning for ontology engineering. *Journal of Web Semantics*, 9:71 – 81, 2011.
82. Jens Lehmann, Chris Bizer, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. DBpedia - a crystallization point for the web of data. *Journal of Web Semantics*, 7(3):154–165, 2009.
83. Jens Lehmann and Lorenz Bühmann. Autosparql: Let users query your knowledge base. In *Proceedings of ESWC 2011*, 2011.
84. Jens Lehmann, Tim Furche, Giovanni Grasso, Axel-Cyrille Ngonga Ngomo, Christian Schallhart, Andrew Sellers, Christina Unger, Lorenz Bühmann, Daniel Gerber, Konrad Höffner, David Liu, and Sören Auer. Deqa: Deep web extraction for question answering. In *Proceedings of ISWC*, 2012.
85. Jens Lehmann and Pascal Hitzler. Foundations of refinement operators for description logics. In Hendrik Blockeel, Jan Ramon, Jude W. Shavlik, and Prasad Tadepalli, editors, *Inductive Logic Programming, 17th International Conference, ILP 2007, Corvallis, OR, USA, June 19-21, 2007*, volume 4894 of *Lecture Notes in Computer Science*, pages 161–174. Springer, 2007. Best Student Paper Award.
86. Jens Lehmann and Pascal Hitzler. A refinement operator based learning algorithm for the $\mathcal{ALC}$ description logic. In Hendrik Blockeel, Jan Ramon, Jude W. Shavlik, and Prasad Tadepalli, editors, *Inductive Logic Programming, 17th International Conference, ILP 2007, Corvallis, OR, USA, June 19-21, 2007*, volume 4894 of *Lecture Notes in Computer Science*, pages 147–160. Springer, 2007. Best Student Paper Award.
87. Jens Lehmann and Pascal Hitzler. Concept learning in description logics using refinement operators. *Machine Learning journal*, 78(1-2):203–250, 2010.
88. David Kopcso Leo Pipino, Ricahrd Wang and William Rybold. *Developing Measurement Scales for Data-Quality Dimensions*, volume 1. M.E. Sharpe, New York, 2005.
89. Bo Leuf and Ward Cunningham. *The Wiki Way: Collaboration and Sharing on the Internet*. Addison-Wesley Professional, 2001.
90. Francesca A. Lisi. Building rules on top of ontologies for the semantic web with inductive logic programming. *Theory and Practice of Logic Programming*, 8(3):271–300, 2008.

91. Francesca A. Lisi and Floriana Esposito. Learning SHIQ+log rules for ontology evolution. In *SWAP 2008*, volume 426 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.
92. Steffen Lohmann, Philipp Heim, Sören Auer, Sebastian Dietzold, and Thomas Riechert. Semantifying requirements engineering – the softwiki approach. In *Proceedings of the 4th International Conference on Semantic Technologies (I-SEMANTICS '08)*, J.UCS, pages 182–185, 2008.
93. Vanessa Lopez, Victoria Uren, Marta Reka Sabou, and Enrico Motta. Cross ontology query answering on the semantic web: an initial evaluation. In *K-CAP '09*, pages 17–24, New York, NY, USA, 2009. ACM.
94. Li Ma, Xingzhi Sun, Feng Cao, Chen Wang, and Xiaoyuan Wang. Semantic enhancement for enterprise data management. In *ISWC2009*, 2009.
95. Michael Martin, Claus Stadler, Philipp Frischmuth, and Jens Lehmann. Increasing the financial transparency of european commission project funding. *Semantic Web Journal*, Special Call for Linked Dataset descriptions, 2013.
96. Y. Matsuo and M. Ishizuka. Keyword Extraction From A Single Document Using Word Co-Occurrence Statistical Information. *International Journal on Artificial Intelligence Tools*, 13(1):157–169, 2004.
97. Brian McBride and Dave Beckett. Rdf/xml syntax specification. W3C Recommendation, February 2004.
98. James McCusker and Deborah McGuinness. Towards identity in linked data. In *Proceedings of OWL Experiences and Directions Seventh Annual Workshop*, 2010.
99. P Mendes, H Mühleisen, and C Bizer. Sieve: Linked data quality assessment and fusion. In *LWDM*, March 2012.
100. Ryan Moats. Urn syntax. Internet RFC 2141, May 1997.
101. Mohamed Morsey, Jens Lehmann, Sören Auer, and Axel-Cyrille Ngonga Ngomo. DBpedia SPARQL Benchmark – Performance Assessment with Real Queries on Real Data. In *ISWC 2011*, 2011.
102. Mohamed Morsey, Jens Lehmann, Sören Auer, and Axel-Cyrille Ngonga Ngomo. Usage-Centric Benchmarking of RDF Triple Stores. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI 2012)*, 2012.
103. Mohamed Morsey, Jens Lehmann, Sören Auer, Claus Stadler, and Sebastian Hellmann. DBpedia and the Live Extraction of Structured Data from Wikipedia. *Program: electronic library and information systems*, 46:27, 2012.
104. D. Nadeau. *Semi-Supervised Named Entity Recognition: Learning to Recognize 100 Entity Types with Little Supervision*. PhD thesis, University of Ottawa, 2007.
105. David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26, January 2007. Publisher: John Benjamins Publishing Company.
106. David Nadeau, Peter Turney, and Stan Matwin. Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity. pages 266–277, 2006.
107. Axel-Cyrille Ngonga Ngomo. Parameter-free clustering of protein-protein interaction graphs. In *Proceedings of Symposium on Machine Learning in Systems Biology 2010*, 2010.
108. Axel-Cyrille Ngonga Ngomo. A time-efficient hybrid approach to link discovery. In *Proceedings of OM@ISWC*, 2011.
109. Axel-Cyrille Ngonga Ngomo. Link discovery with guaranteed reduction ratio in affine spaces with minkowski measures. In *Proceedings of ISWC*, 2012.
110. Axel-Cyrille Ngonga Ngomo. On link discovery using a hybrid approach. *Journal on Data Semantics*, 1:203 – 217, December 2012.
111. Axel-Cyrille Ngonga Ngomo and Sören Auer. Limes - a time-efficient approach for large-scale link discovery on the web of data. In *Proceedings of IJCAI*, 2011.

112. Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann, and Daniel Gerber. Sorry, i don't speak sparql — translating sparql queries into natural language. In *Proceedings of WWW*, 2013.
113. Axel-Cyrille Ngonga Ngomo, Lars Kolb, Norman Heino, Michael Hartung, Sören Auer, and Erhard Rahm. When to reach for the cloud: Using parallel hardware for link discovery. In *Proceedings of ESCW*, 2013.
114. Axel-Cyrille Ngonga Ngomo, Jens Lehmann, Sören Auer, and Konrad Höffner. Raven – active learning of link specifications. In *Proceedings of OM@ISWC*, 2011.
115. Axel-Cyrille Ngonga Ngomo and Klaus Lyko. Eagle: Efficient active learning of link specifications using genetic programming. In *Proceedings of ESWC*, 2012.
116. Axel-Cyrille Ngonga Ngomo, Klaus Lyko, and Victor Christen. Coala – correlation-aware active learning of link specifications. In *Proceedings of ESWC*, 2013.
117. Axel-Cyrille Ngonga Ngomo and Frank Schumacher. Border flow – a local graph clustering algorithm for natural language processing. In *Proceedings of the 10th International Conference on Intelligent Text Processing and Computational Linguistics (CICLING 2009)*, pages 547–558, 2009. Best Presentation Award.
118. Dat P. T. Nguyen, Yutaka Matsuo, and Mitsuru Ishizuka. Relation extraction from wikipedia using subtree mining. In *AAAI*, pages 1414–1420, 2007.
119. Thuy Nguyen and Min-Yen Kan. Keyphrase Extraction in Scientific Publications. pages 317–326. 2007.
120. Shan-Hwei Nienhuys-Cheng and Ronald de Wolf, editors. *Foundations of Inductive Logic Programming*, volume 1228 of *LNCS*. Springer, 1997.
121. Eyal Oren. SemperWiki: A Semantic Personal Wiki. In Stefan Decker, Jack Park, Dennis Quan, and Leo Sauermann, editors, *Proc. of Semantic Desktop Workshop at the ISWC, Galway, Ireland, November 6*, volume 175, November 2005.
122. Patrick Pantel and Marco Pennacchiotti. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *ACL*, pages 113–120. ACL Press, 2006.
123. Youngja Park, Roy J Byrd, and Branimir K Boguraev. Automatic glossary extraction: beyond terminology identification. In *Proceedings of the 19th international conference on Computational linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.
124. Marius Pasca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. Organizing and searching the world wide web of facts - step one: the one-million fact extraction challenge. In *proceedings of the 21st national conference on Artificial intelligence - Volume 2*, pages 1400–1405. AAAI Press, 2006.
125. Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. OWL Web Ontology Language - Semantics and Abstract Syntax. W3c:rec, W3C, 10 feb 2004. `http://www.w3.org/TR/owl-semantics/`.
126. Erhard Rahm. *Schema Matching and Mapping*. Springer-Verlag, Heidelberg (DE), 2011.
127. Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10:334–350, 2001.
128. Yves Raimond, Christopher Sutton, and Mark Sandler. Automatic interlinking of music datasets on the semantic web. In *1st Workshop about Linked Data on the Web*, 2008.
129. Thomas Riechert, Kim Lauenroth, Jens Lehmann, and Sören Auer. Towards semantic based requirements engineering. In *Proceedings of the 7th International Conference on Knowledge Management (I-KNOW)*, 2007.
130. Thomas Riechert, Ulf Morgenstern, Sören Auer, Sebastian Tramp, and Michael Martin. Knowledge engineering for historians on the example of the catalogus professorum lipsiensis. In Peter F. Patel-Schneider, Yue Pan, Pascal Hitzler, Peter Mika, Lei Zhang, Jeff Z. Pan, Ian Horrocks, and Birte Glimm, editors, *Proceedings of the 9th International Semantic*

Web Conference (ISWC2010), volume 6497 of *Lecture Notes in Computer Science*, pages 225–240, Shanghai / China, 2010. Springer.

131. Christoph Rieß, Norman Heino, Sebastian Tramp, and Sören Auer. EvoPat – Pattern-Based Evolution and Refactoring of RDF Knowledge Bases. In *Proceedings of the 9th International Semantic Web Conference (ISWC2010)*, Lecture Notes in Computer Science, Berlin / Heidelberg, 2010. Springer.
132. Sebastian Rudolph. Exploring relational structures via FLE. In Karl Erich Wolff, Heather D. Pfeiffer, and Harry S. Delugach, editors, *ICCS 2004*, volume 3127 of *LNCS*, pages 196–212. Springer, 2004.
133. Anisa Rula, Matteo Palmonari, Andreas Harth, Steffen Stadtmüller, and Andrea Maurino. On the diversity and availability of temporal information in linked open data. In *ISWC*, 2012.
134. Satya S. Sahoo, Wolfgang Halb, Sebastian Hellmann, Kingsley Idehen, Ted Thibodeau Jr, Sören Auer, Juan Sequeda, and Ahmed Ezzat. A survey of current approaches for mapping of relational databases to rdf, 01 2009.
135. G. Sampson. How fully does a machine-usable dictionary cover english text. *Literary and Linguistic Computing*, 4(1), 1989.
136. Leo Sauermann and Richard Cyganiak. Cool uris for the semantic web. W3C Interest Group Note, December 2008.
137. Sebastian Schaffert. Ikewiki: A semantic wiki for collaborative knowledge management. In *Proceedings of the 1st International Workshop on Semantic Technologies in Collaborative Applications (STICA)*, 2006.
138. Francois Scharffe, Yanbin Liu, and Chuguang Zhou. Rdf-ai: an architecture for rdf datasets matching, fusion and interlink. In *Proc. IJCAI 2009 IR-KR Workshop*.
139. Baris Sertkaya. OntocomP system description. In Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, and Ulrike Sattler, editors, *Proceedings of the 22nd International Workshop on Description Logics (DL 2009), Oxford, UK, July 27-30, 2009*, volume 477 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.
140. Burr Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.
141. Pavel Shvaiko and Jerome Euzenat. Ten challenges for ontology matching. Technical report, August 01 2008.
142. Adam Souzis. Building a Semantic Wiki. *IEEE Intelligent Systems*, 20(5):87–91, 2005.
143. Dimitrios-Emmanuel Spanos, Periklis Stavrou, and Nikolas Mitrou. Bringing relational databases into the semantic web: A survey. *Semantic Web*, 3(2):169–209, 2012.
144. Claus Stadler, Jens Lehmann, Konrad Höffner, and Sören Auer. Linkedgeodata: A core for a web of spatial open data. *Semantic Web Journal*, 3(4):333–354, 2012.
145. Christine Thielen. An approach to proper name tagging for german. In *In Proceedings of the EACL-95 SIGDAT Workshop*, 1995.
146. Sebastian Tramp, Philipp Frischmuth, Timofey Ermilov, and Sören Auer. Weaving a Social Data Web with Semantic Pingback. In P. Cimiano and H.S. Pinto, editors, *Proceedings of the EKAW 2010 - Knowledge Engineering and Knowledge Management by the Masses; 11th October-15th October 2010 - Lisbon, Portugal*, volume 6317 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 135–149, Berlin / Heidelberg, October 2010. Springer.
147. Sebastian Tramp, Norman Heino, Sören Auer, and Philipp Frischmuth. RDFauthor: Employing RDFa for collaborative Knowledge Engineering. In P. Cimiano and H.S. Pinto, editors, *Proceedings of the EKAW 2010 - Knowledge Engineering and Knowledge Management by the Masses; 11th October-15th October 2010 - Lisbon, Portugal*, volume 6317 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 90–104, Berlin / Heidelberg, October 2010. Springer.

148. Peter D. Turney. Coherent keyphrase extraction via web mining. In *Proceedings of the 18th international joint conference on Artificial intelligence*, pages 434–439, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.
149. Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. Sparql template-based question answering. In *Proceedings of WWW*, 2012.
150. Jacopo Urbani, Spyros Kotoulas, Jason Maassen, Frank van Harmelen, and Henri Bal. Owl reasoning with webpie: calculating the closure of 100 billion triples. In *ESWC2010*, 2010.
151. Mateja Verlic. Lodgrefine - lod-enabled google refine in action. In *I-SEMANTICS 2012 Posters and Demonstrations Track*, pages 31 – 27, 2012.
152. Johanna Völker and Mathias Niepert. Statistical schema induction. In *ESWC (1)*, pages 124–138, 2011.
153. Johanna Völker and Sebastian Rudolph. Fostering web intelligence by semi-automatic OWL ontology refinement. In *Web Intelligence*, pages 454–460. IEEE, 2008.
154. Johanna Völker, Denny Vrandecic, York Sure, and Andreas Hotho. Learning disjointness. In *ESWC 2007*, volume 4519 of *LNCS*, pages 175–189. Springer, 2007.
155. Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Discovering and maintaining links on the web of data. In *ISWC 2009*, pages 650–665. Springer, 2009.
156. D. Walker and R. Amsler. The use of machine-readable dictionaries in sublanguage analysis. *Analysing Language in Restricted Domains*, 1986.
157. Gang Wang, Yong Yu, and Haiping Zhu. Pore: Positive-only relation extraction from wikipedia text. In *ISWC07*, volume 4825 of *LNCS*, pages 575–588, Berlin, Heidelberg, 2007. Springer Verlag.
158. Richard Y. Wang and Diane M. Strong. Beyond accuracy: what data quality means to data consumers. *Journal of Management Information Systems*, 12(4):5–33, 1996.
159. Hiroaki Watanabe and Stephen Muggleton. Can ILP be applied to large dataset? In *ILP 2009*, 2009.
160. William Winkler. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Bureau of the Census, 1999.
161. William Winkler. Overview of record linkage and current research directions. Technical report, Bureau of the Census - Research Report Series, 2006.
162. Harris Wu, Mohammad Zubair, and Kurt Maly. Harvesting social knowledge from folksonomies. In *Proceedings of the seventeenth conference on Hypertext and hypermedia*, HYPERTEXT '06, pages 111–114, New York, NY, USA, 2006. ACM.
163. Yulan Yan, Naoaki Okazaki, Yutaka Matsuo, Zhenglu Yang, and Mitsuru Ishizuka. Unsupervised relation extraction by mining wikipedia texts using information from the web. In *ACL*, ACL '09, pages 1021–1029, 2009.
164. Amrapali Zaveri, Jens Lehmann, Sören Auer, Mofeed M. Hassan, Mohamed A. Sherif, and Michael Martin. Publishing and interlinking the global health observatory dataset. *Semantic Web Journal*, Special Call for Linked Dataset descriptions, 2013.
165. Amrapali Zaveri, Ricardo Pietrobon, Sören Auer, Jens Lehmann, Michael Martin, and Timofey Ermilov. Redd-observatory: Using the web of data for evaluating the research-disease disparity. In *Proc. of the IEEE/WIC/ACM International Conference on Web Intelligence*, 2011.
166. Amrapali Zaveri, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann, and Sören Auer. Quality assessment methodologies for linked open data. Under review, available at http://www.semantic-web-journal.net/content/quality-assessment-methodologies-linked-open-data.
167. GuoDong Zhou and Jian Su. Named entity recognition using an hmm-based chunk tagger. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*,

ACL '02, pages 473–480, Morristown, NJ, USA, 2002. Association for Computational Linguistics.