D3.8.1 Prototype for Learning Networked Ontologies

Deliverable Co-ordinator: **Johanna Völker**

Deliverable Co-ordinating Institution: **University of Karlsruhe (UKARL)**

Other Authors: **Johanna Völker, Eva Blomqvist**

With Contributions from: **Christian Meilicke and Heiner Stuckenschmidt (University of Mannheim), Sebastian Rudolph (UKARL)**

In this deliverable, we present a first set of methods and tools for the acquisition, evaluation and refinement of networked ontologies. We argue that context plays an important role for each of these tools and present initial ideas with regards to an integrated framework of semi-automatic ontology engineering.

Document Identifier:	NEON/2008/D3.8.1/v1.3	Date due:	February 29, 2008
Class Deliverable:	NEON EU-IST-2005-027595	Submission date:	February 29, 2008
Project start date	March 1, 2006	Version:	v1.3
Project duration:	4 years	State:	Final
		Distribution:	Public

NeOn Consortium

This document is part of the NeOn research project funded by the IST Programme of the Commission of the European Communities by the grant number IST-2005-027595. The following partners are involved in the project:

Open University (OU) – Coordinator	**Universität Karlsruhe – TH (UKARL)**
Knowledge Media Institute – KMi	Institut für Angewandte Informatik und Formale
Berrill Building, Walton Hall	Beschreibungsverfahren – AIFB
Milton Keynes, MK7 6AA	Englerstrasse 11
United Kingdom	D-76128 Karlsruhe, Germany
Contact person: Martin Dzbor, Enrico Motta	Contact person: Peter Haase
E-mail address: {m.dzbor, e.motta}@open.ac.uk	E-mail address: pha@aifb.uni-karlsruhe.de
Universidad Politécnica de Madrid (UPM)	**Software AG (SAG)**
Campus de Montegancedo	Uhlandstrasse 12
28660 Boadilla del Monte	64297 Darmstadt
Spain	Germany
Contact person: Asunción Gómez Pérez	Contact person: Walter Waterfeld
E-mail address: asun@fi.ump.es	E-mail address: walter.waterfeld@softwareag.com
Intelligent Software Components S.A. (ISOCO)	**Institut 'Jožef Stefan' (JSI)**
Calle de Pedro de Valdivia 10	Jamova 39
28006 Madrid	SL–1000 Ljubljana
Spain	Slovenia
Contact person: Jesús Contreras	Contact person: Marko Grobelnik
E-mail address: jcontreras@isoco.com	E-mail address: marko.grobelnik@ijs.si
Institut National de Recherche en Informatique et en Automatique (INRIA)	**University of Sheffield (USFD)**
ZIRST – 665 avenue de l'Europe	Dept. of Computer Science
Montbonnot Saint Martin	Regent Court
38334 Saint-Ismier, France	211 Portobello street
Contact person: Jérôme Euzenat	S14DP Sheffield, United Kingdom
E-mail address: jerome.euzenat@inrialpes.fr	Contact person: Hamish Cunningham
	E-mail address: hamish@dcs.shef.ac.uk
Universität Kolenz-Landau (UKO-LD)	**Consiglio Nazionale delle Ricerche (CNR)**
Universitätsstrasse 1	Institute of cognitive sciences and technologies
56070 Koblenz	Via S. Marino della Battaglia
Germany	44 – 00185 Roma-Lazio Italy
Contact person: Steffen Staab	Contact person: Aldo Gangemi
E-mail address: staab@uni-koblenz.de	E-mail address: aldo.gangemi@istc.cnr.it
Ontoprise GmbH. (ONTO)	**Food and Agriculture Organization of the United Nations (FAO)**
Amalienbadstr. 36	Viale delle Terme di Caracalla
(Raumfabrik 29)	00100 Rome
76227 Karlsruhe	Italy
Germany	Contact person: Marta Iglesias
Contact person: Jürgen Angele	E-mail address: marta.iglesias@fao.org
E-mail address: angele@ontoprise.de	
Atos Origin S.A. (ATOS)	**Laboratorios KIN, S.A. (KIN)**
Calle de Albarracín, 25	C/Ciudad de Granada, 123
28037 Madrid	08018 Barcelona
Spain	Spain
Contact person: Tomás Pariente Lobo	Contact person: Antonio López
E-mail address: tomas.parientelobo@atosorigin.com	E-mail address: alopez@kin.es

Work package participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed to the writing of this document or its parts:

- Institute AIFB, University of Karlsruhe (UKARL)

- Consiglio Nazionale delle Ricerche (CNR)

Change Log

Version	Date	Amended by	Changes
0.1	29-09-2007	Johanna Völker	Initial setup
0.2	10-01-2008	Johanna Völker	First draft of Section 4
0.3	13-01-2008	Johanna Völker	First draft of Sections 2 and 3
0.4	16-01-2008	Eva Blomqvist	First draft of Section 5
0.5	25-01-2008	Johanna Völker	Revision of Sections 2-4
0.6	26-01-2008	Johanna Völker	First draft of Section 1
0.7	29-01-2008	Eva Blomqvist	Revision of Section 5
0.8	30-01-2008	Johanna Völker	First draft of Section 6
0.9	31-01-2008	Eva Blomqvist	Executive summary
1.0	02-02-2008	Johanna Völker	Final additions and changes
1.1	28-02-2008	Johanna Völker	Update of Sections 2.4 and 3.5
1.2	28-02-2008	Eva Blomqvist	Revision of Sections 1 and 5
1.3	10-04-2008	Johanna Völker	Minor changes to front page

Executive Summary

Reasoning-based applications are revolutionizing research in many complex domains and the ontologies employed by these applications are required to be of great expressivity, quality and size. Consequently, building such ontologies is a difficult and time-consuming task. Also the sheer size of these ontologies (and knowledge bases) makes their manual evaluation and refinement an extremely difficult endeavor. In order to overcome these problems, tools for semi-automatic ontology refinement should support the user not only in ontology construction but also in the process of evaluating and refining the acquired axiomatizations. A complementary way of assisting the user in the ontology engineering process is to introduce knowledge reuse by means of patterns, i.e. template solutions representing best-practises of the community. In this deliverable, we present several methods for the acquisition, refinement and evaluation of expressive ontologies. The deliverable first relates semi-automatic ontology construction and refinement to definition of context, and then a set of methods and tools for learning ontologies are described in detail. Additionally the framework for pattern-based ontology refinement is presented and alignment to top-level ontologies is discussed. The deliverable additionally present the overall view in NeOn on how these techniques assist the user in constructing ontologies and putting learned ontologies into a context, additionally implementations, and planned implementations, of these approaches as parts of the NeOn Toolkit are considered.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Intelligent, reasoning-based applications are well on the way to revolutionize research in complex domains such bioinformatics and medicine. The ontologies employed by these applications are required to be of great expressivity, quality and size in order to enable the discovery of new knowledge and thus important scientific advances. Consequently, building such ontologies is a difficult and time-consuming task, requiring to combine the knowledge of highly specialized domain experts with the skill and experience of ontology engineers, hence resulting in a high demand on scarce expert resources. At the same time, the sheer size of these ontologies (respectively, knowledge bases) makes their manual evaluation and refinement an extremely difficult endeavor.

In order to overcome these problems, tools for semi-automatic ontology refinement should support the user not only in ontology construction, i.e. in specifying her conceptualization of the domain, but also in the process of evaluating and refining the acquired axiomatizations. The first requirement can be efficiently fulfilled by automated ontology acquisition methods that extract specifications of potentially relevant knowledge (complex domain axioms or facts) from domain-specific resources. These methods must be complemented by ontology evaluation techniques to assure logical completeness, correct model-theoretic semantics or formal and logical consistency of the lexically acquired ontologies.

A complementary way of assisting the user in the ontology engineering process is to introduce knowledge reuse in semi-automatic ontology construction. A common way to introduce reuse in other areas is through the use of patterns, template solutions representing best-practises of the community. Such ontology engineering patterns are already used in manual methods for ontology design, but so far few approaches use patterns in a semi-automatic way. There are many challenges when applying patterns semi-automatically, for example how to match learned ontologies to the patterns and bridge the abstraction gap between general patterns and specific terms and relations extracted from domain specific texts. This is a challenge closely related to ontology matching and ontology search and ranking, but expose some specific characteristics to guide the methods. Other challenges lie in pattern specialization and composition, in order to produce a coherent and correct ontology. Additional evaluation and revision is also needed to ensure that the patterns applied semi-automatically produce a reasonable ontology. Apart from these challenges, the use of patterns also give many benefits, one specific benefit being the alignment of learned ontologies to top-level ontologies, since many patterns have been extracted from top-level ontologies and can thereby provide a link back to this origin.

In this deliverable, we present several methods for the acquisition, refinement and evaluation of expressive ontologies. We start by a short overview of NeOn, focusing on WP3 and the definition of context (cf. Section 1.1), which we will occasionally refer to in the remainder of this deliverable. In Chapter 2, we describe the ontology learning framework Text2Onto and a plugin that we developed to make its basic ontology learning functionality available as part of the NeOn Toolkit. Chapter 3 introduces LeDA, a tool for the automatic generation of disjointness axioms, developed to support both enrichment and evaluation of the acquired ontologies. As for Text2Onto, we describe its relevance with respect to our definition of context and present a corresponding Toolkit plugin. In Chapter 4, we describe a new approach to interactive ontology refinement,

developed to ontology engineers in the acquisition of complex class descriptions and subsumption relationships. Subsequently in chapter 5 the OntoCase approach, a framework for further refinement of learned ontologies by means of ontology patterns, is presented along with the benefits this can give through aligning learnt ontologies to top-level ontologies. OntoCase is related to the ontology pattern activities in WP2 and WP5 and has originated in this work, but OntoCase presents a completely new approach that connects ontology learning and the use of patterns, and introduces interesting context-related aspects when connecting learnt ontologies to patterns and possibly also top-level ontologies. Finally, Chapter 6 concludes with a summary and an outlook to future work.

1.1   The Big Picture: NeOn and WP3

The research reported in this deliverable is part of NeOn WP3 – "Context sensitivity for Networked Ontologies". As illustrated by Figure 1.1, WP3 constitutes a central component of the overall NeOn architecture. Among the most important goals of this workpackage is the development of formalisms and methods for dealing with context-sensitivity of ontologies, including specific solutions for applications such as ontology alignment or reasoning.
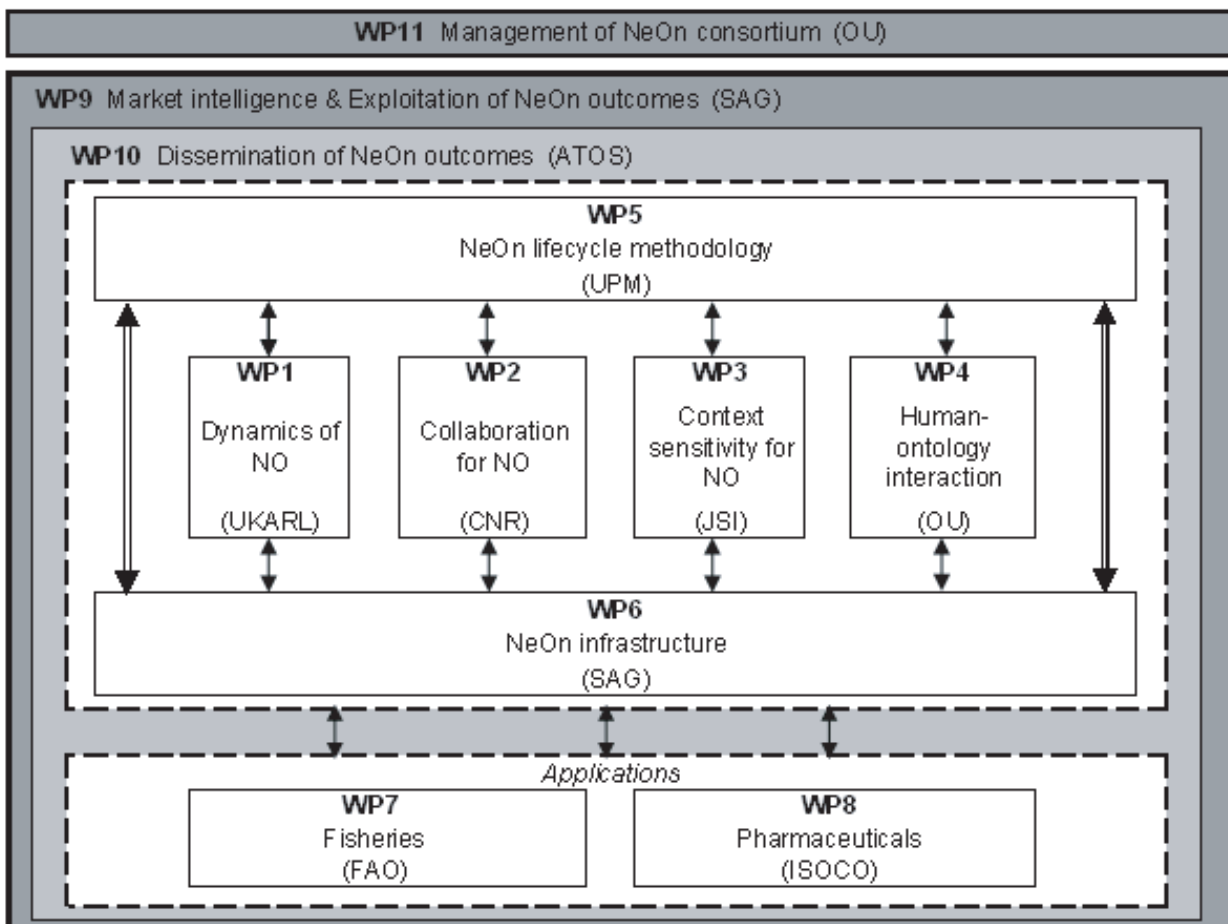
Figure 1.1: Relationships between different workpackages in NeOn

Context sensitivity for networked ontologies.   For this deliverable, we adopt the notion of context as a *semantic modifier*, which is considered something that changes our interpretation of a knowledge base. This

definition was first introduced in NeOn D3.1.1 and refined later by further definitions, e.g., in NeOn D3.1.3. The latter deliverable instantiates the original abstract notion of context in three different ways: *provenance*, *argumentation* and *mapping*.

- *Provenance* refers to the origin of an ontology element and constitutes the most important type of context in this deliverable. By associating an entity or axiom with provenance information we can represent, for example, knowledge about its creator (typically, a human or automatic agent), the time it was added to the ontology or lexical resources providing a justification for its existence.

- *Mappings* are directed semantic relationships (also called "correspondences") between elements in different ontologies. As suggested in D3.1.3, *un*directed correspondences can be represented by bi-directional mappings. In the following, we will use the term *ontology alignment* to refer to a set of correspondences including, e.g., subsumption or equivalence relationships.

- *Argumentation* is the exchange of arguments in favor or against particular ontology design decisions.

1.2 Acknowledgements

Chapter 2

Basic Ontology Learning

Text2Onto[1] is an ontology learning framework, originally developed in the SEKT project[2]. In this chapter, we present a new graphical frontend for Text2Onto implemented as a tightly coupled GUI plugin for the NeOn toolkit.

Section 2.1 reviews the notion of context from an ontology learning perspective, followed by a brief overview of Text2Onto 2.2 and the ontology learning methods it provides 2.3. In Section 2.4, we describe in detail the new Text2Onto plugin for the NeOn toolkit before concluding in Section 2.5.

2.1　Context Sensitivity for Networked Ontologies

If we recall the definition of context as a semantic modifier (see also Section 1.1), we will recognize in what follows, at least two types of context.

Lexical resources.　The lexical resources analysed by Text2Onto – in particular, natural language corpora, WordNet [Fel98] or the World Wide Web – constitute a semantic modifier for the learned ontology insofar as they provide a justification for the inclusion of entities and axioms in this ontology. As the corpus changes over time, e.g., according to changing user interests or application requirements, the incremental ontology learning methods of Text2Onto[3] will be likely to suggest the addition or removal of particular ontology elements. In this sense, the evidence obtained from various lexical resources must be considered as a kind context that shapes the content and our interpretation of the ontology. Furthermore, it also influences our perception of quality when it comes to the question how suitable an ontology is for a particular application or whether it is still up-to-date with regards to the domain knowledge represented by a corpus [BADW04].

Provenance information.　As detailed in Section 2.3, Text2Onto's *Model of Possible Ontologies* associates various types of provenance information with the learned ontology elements. Each ontology learning method generates confidence or relevance values, respectively, for the entities and axioms generated from the lexical resources. Whereas the confidence values represent the certainty of the methods with respect to the correctness of a particular result, relevance values indicate how characteristic certain elements are for the domain of interest. Besides these two kinds of certainty values, a change history is maintained by the ontology management back-end of Text2Onto. It comprises detailed information about the method, source and time of an element's creation.

---

[1]http://ontoware.org/projects/text2onto/
[2]http://www.sekt-project.com/
[3]For more detailed explanations with respect to the process of incremental ontology learning and *data-driven change discovery*, please refer to [VS05].

2.2   Text2Onto

Text2Onto [CV05a] is an ontology learning framework which has been developed to support the acquisition of ontologies from textual documents. Like its predecessor, TextToOnto [MV01], it provides an extensible set of methods for learning atomic classes, class subsumption and instantiation as well as object properties and disjointness axioms.

The core of Text2Onto is the so-called *Model of Possible Ontologies* (POM), a collection of instantiated modeling primitives, which are independent of a concrete ontology representation language. Every instance of one of these modeling primitives gets assigned a number of rating annotations indicating how certain the algorithm in question is about the correctness or relevance of the corresponding instance. The purpose of this type of provenance information is to facilitate the user interaction by allowing her to filter the POM and thereby select only a number of relevant instances of modeling primitives to be translated into a target language of her choice.

Text2Onto's POM can be populated by means of several methods based on natural language processing and machine learning techniques. In the following section, we describe some of these methods in more detail.

2.3   Ontology Learning Methods

Concepts and Instances.   Different term weighting measures are used to compute the relevance of a certain concept or instance with respect to the corpus: Relative Term Frequency, TFIDF [Sal91], Entropy [Gra90] and the C-value/NC-value method in [KF98]. For each term, the values of these measures are normalized into the interval $[0..1]$ and used as corresponding relevance in the POM.

Subclass-of Relations.   In order to learn subclass-of relations, we have implemented a variety of different algorithms exploiting the hypernym structure of WordNet [Fel98], matching Hearst patterns [Hea92] in the corpus as well as in the WWW and applying linguistic heuristics mentioned in [VNCN05]. The resulting confidence values of these algorithms are then combined through combination strategies as described in [CPSTS05].

Instance-of Relations.   In order to assign instances or named entities appearing in the corpus to a concept in the ontology Text2Onto relies on a similarity-based approach extracting context vectors for instances and concepts from the text collection and assigning instances to the concept corresponding to the vector with the highest similarity with respect to their own vector [CV05b]. Alternatively, we also implemented a pattern-matching algorithm similar to the one used for discovering part-of relations.

General Relations.   To learn general relations, Text2Onto employs a shallow parsing strategy to extract subcategorization frames (e.g. `hit(subj,obj,pp(with))`, transitive + PP-complement) enriched with information about the frequency of the terms appearing as arguments [MS00]. These subcategorization frames are mapped to relations such as `hit`(*person*,*thing*) and `hit_with`(*person*,*object*). The confidence is estimated on the basis of the frequency of the subcategorization frame as well as of the frequency with which a certain term appears at the argument position.

Equivalence and Equality.   Following the assumption that terms are similar to the extent to which they share similar syntactic contexts, we implemented algorithms calculating the similarity between terms on the basis of contextual features extracted from the corpus, whereby the context of a terms varies from simple word windows to linguistic features extracted with a shallow parser. This corpus-based similarity is then taken as the confidence for the equivalence of the corresponding concepts or instances.

Disjointness. For the extraction of disjointness axioms we implemented a simple heuristic based on lexico-syntactic patterns. In particular, given an enumeration of noun phrases $NP_1, NP_2, ...(and|or)NP_n$ we conclude that the concepts $C_1, C_2, ...C_k$ denoted by these noun phrases are pairwise disjoint, where the confidence for the disjointness of two concepts is obtained from the number of evidences found for their disjointness in relation to the total number of evidences for the disjointness of these concepts with other concepts.

A more effective approach to learning disjointness axioms is implemented in the LeDA framework. For details, please refer to Chapter 3.

Subtopic-of Relations. In order to identify subtopic-of relationships, we implemented an approach by [SC99]. It is based on the assumption that each topic tends to occur in a true subset of all the documents containing its supertopic. Therefore, it creates subtopic-of relationships by considering inclusion relationships between the document sets associated with all concepts in the ontology. Additionally, we also developed a very simple algorithm that generates subtopic-of relation from previously extracted subclass-of relationships.

2.4   NeOn Toolkit Plugin

In NeOn, we developed a graphical frontend for Text2Onto that will be made available as a tightly coupled GUI plugin for the NeOn toolkit. The plugin, which constitutes a core part of our protoype for learning networked ontologies, will enable the integration of Text2Onto into a process of semi-automatic ontology engineering. Sources and binaries of the Text2Onto plugin can be obtained from Ontoware[4] or the NeOn plugin repository.

2.4.1   User Guide

In the following, we provide a short user guide, that is intended to help ontology engineers to get started with the Text2Onto plugin. For detailed installation instructions please refer to Section 2.4.2.

The graphical user interface of the plugin (cf. Figure 2.1) is very similar to the original Swing-based GUI of Text2Onto.[5] It is composed of different views for the configuration of the ontology learning process and the presentation of the results.

Workflow view. The upper left corner contains the *workflow view*, which is used to set up the ontology learning workflow. By right-clicking on the individual ontology learning tasks (e.g. *Concept*, see also Section 2.3), the user can select one or more methods for each type of ontology element she wants to extract from the corpus.

Corpus view. In the bottom left corner, the user will find a *corpus view*, which allows her to set up a corpus, that is a collection of text documents[6] from which the ontology will be generated. The *doc view* (see hidden tab on the right) is used to display previews of selected documents.

POM view. The *POM view* on the right shows the results of the most recently initiated ontology learning process. The view contains several tabs – one for each type of ontology element that was extracted from the corpus – showing a tabular listing of individual results. By clicking on the column headers the user can sort the ontology elements according to their associated labels or confidence values.

---

[4]http://ontoware.org/projects/text2onto/

[5]Some documentation material and a few demo videos can be downloaded from http://www.aifb.uni-karlsruhe.de/WBS/jvo/text2onto/.

[6]Text2Onto is able to analyse documents in plain text, PDF (Windows only) and HTML format. However, a manual conversion into purely textual format is highly recommended for efficiency reasons.
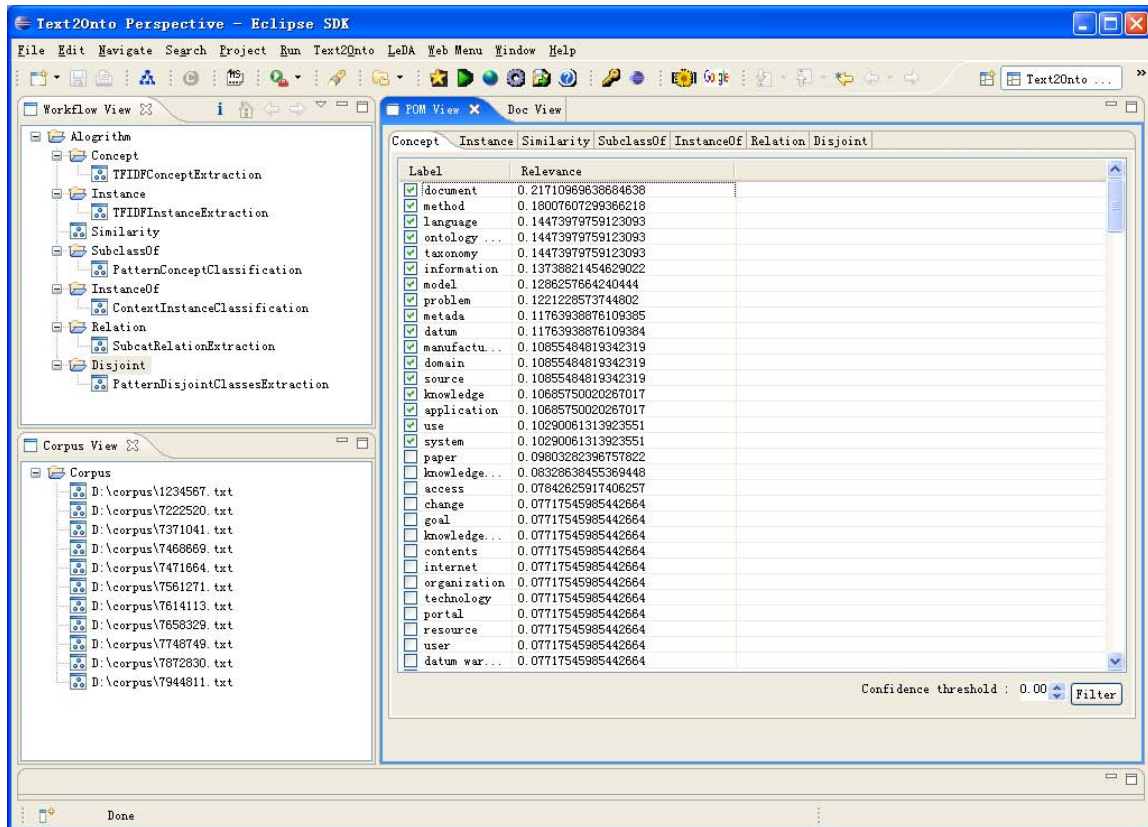
Figure 2.1: Text2Onto plugin: presentation of results
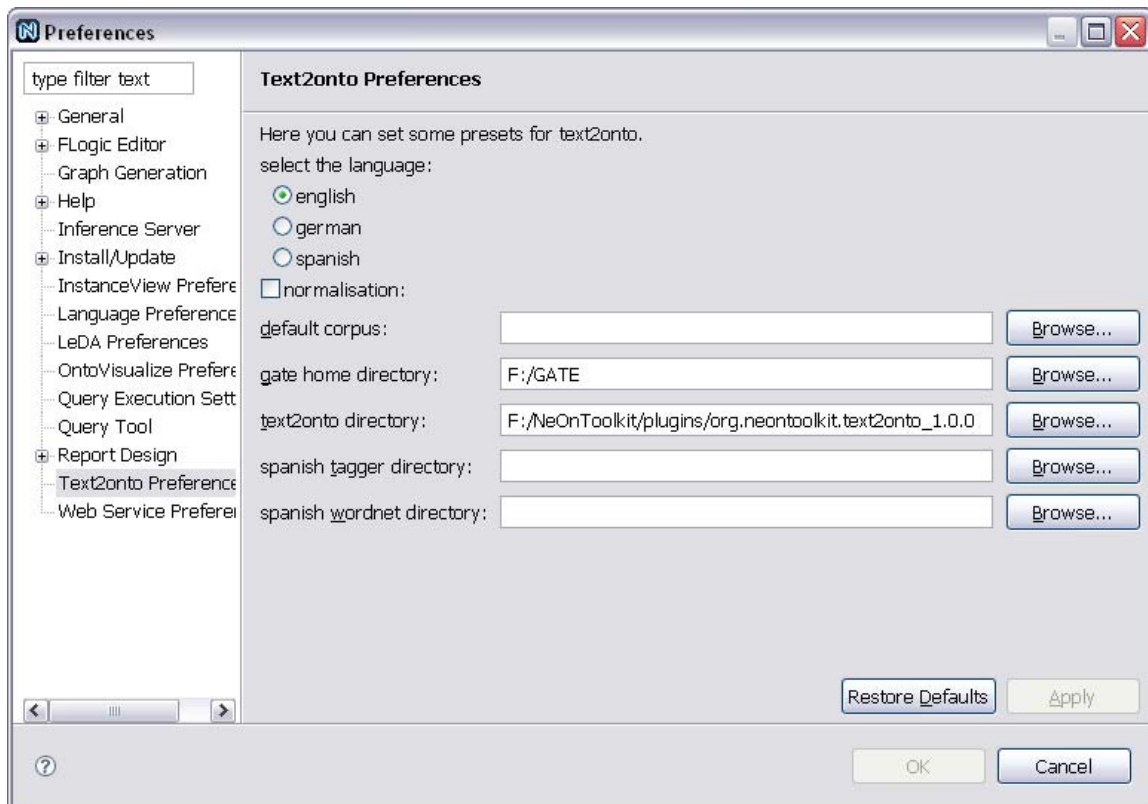
Figure 2.2: Text2Onto plugin: configuration

Preferences. The *preference page*, which is accessible from the main menu of on the top of the Text2Onto perspective (*Window → Preferences... → Text2Onto Preferences*) replaces the original configuration file of Text2Onto's API. As shown by Figure 2.2, it allows for setting various parameters:

- **Language:** The language of the documents to be analysed. Text2Onto provides full support for learning ontologies from *English* and *Spanish* corpora as well as partial support for ontology extraction from *German* texts. For details with respect to the Spanish version of Text2Onto please refer to [VVS07].

- **Normalization:** If this parameter is selected Text2Onto will normalize all confidence values to an interval of 0.0 to 1.0.

- **Default corpus:** The default directory for populating the ontology learning corpus.

- **Spanish tagger directory:** The part-of-speech tagger to be used for the analysis of Spanish documents. In the current version of Text2Onto this parameter is expected to point to the TreeTagger[7] installation directory.

- **Spanish WordNet directory:** In case the language is set to *Spanish*, this path should refer to a licensed version of Spanish WordNet.[8]

2.4.2 Installation

1. Install **GATE 4.0**[9] to <GATE-DIR> (e.g. *c:\GATE*)

2. Install **WordNet 2.0**[10] to <WN-DIR> (e.g. *c:\WordNet*)

3. Unzip **org.neon.toolkit.text2onto_1.0.0.jar** into your Eclipse plugin directory (e.g. <T2O-DIR>=*c:\Eclipse\plugins\org.neon.toolkit.text2onto_1.0.0*)

4. Edit <T2O-DIR>\lib\jwnl**file_properties.xml** and replace <WN-DIR>

```
<param name="file_manager" value="net.didion.jwnl.dictionary.file_manager.
    FileManagerImpl">
    <param name="file_type" value="net.didion.jwnl.princeton.file.
        PrincetonRandomAccessDictionaryFile"/>
    <param name="dictionary_path" value="<WN-DIR>\dict"/>
</param>
```

5. Start NeOn Toolkit (*startup.jar*) and open Text2Onto perspective

6. Set the preferences as described in Section 2.4.1

7. Initialize the algorithm controller by selecting the *New* item from the Text2Onto menu or toolbar

2.5 Conclusion

In this chapter, we gave a short overview of the ontology learning framework Text2Onto. We argued that Text2Onto instantiates our notion of context in various ways finally presented a new graphical frontend for Text2Onto, that will enable the integration of its ontology learning methods into the NeOn toolkit and the overall ontology lifecycle.

---

[7]http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/
[8]http://www.lsi.upc.edu/żnlp/web/index.php
[9]http://gate.ac.uk/download/index.html
[10]http://wordnet.princeton.edu

Chapter 3

Learning Disjointness Axioms

Lightweight ontologies are error-prone in that serious modeling flaws often remain undetected. Since the lack of expressivity restricts the ontological commitment of such ontologies improper modeling rarely leads to contradictory information, which could be detected by automatic or even non-automatic means. In this chapter, we therefore present a method for enriching lightweight ontologies by disjointness axioms. We show that learning disjointness can help to facilitate the task of ontology alignment and present a plugin for the NeOn toolkit.

After a short discussion of the relationship between learning disjointness and the notion of context (see Section 3.1), we describe in detail our approach to the automatic acquisition of disjointness axioms. Section 3.2 introduces LeDA, an open-source framework for learning disjointness, which is based on machine learning classification and uses a variety of different methods to obtain evidence from lexical and logical resources (see Section 3.3). In Section 3.4, we present a thorough evaluation of LeDA and demonstrate the usefulness of our approach within an ontology alignment scenario. The graphical frontend of LeDA, a GUI plugin for the NeOn toolkit, is described in Section 3.5. Section 3.6 concludes with a short summary.

3.1 Context Sensitivity for Networked Ontologies

Our approach to learning disjointness relates to the context-sensitivity of ontologies in several ways.

Lexical and logical resources.   Like Text2Onto (see Section 2), LeDA relies on heterogeneous sources of evidence to accomplish a particular ontology learning task. Each of the lexical and logical resources corresponds to one or more features, that are an essential part of LeDA's classification model. Together with a sufficient amount of training data, i.e. manually labeled pairs of classes, these features enable a reliable, statistical prediction of disjointness relationships.

Uncertainty values.   Based on the probability distribution of the machine learning classifier – NaiveBayes, in our experiments – we yield certainty values, that indicate the estimated reliability of all disjointness axioms. This sort of provenance information represents context insofar as it effects the level of confidence that applications or humans may have with regards to the correctness of the learned axioms.

Direct mappings.   Disjointness axioms can be seen as a special type of direct correspondence between two classes, hence constituting a natural kind of contextualization. Although our most recent experiments focus on the generation of axioms within one ontology, our approach is applicable to scenarios, where disjointness axioms are required to relate classes in different ontologies. At least, the lexical features – in particular those based on label similarity, WordNet and automatically generated background knowledge (see Section 3.3) – are independent of a formal semantic relationship between the respective pairs of classes.

Mapping debugging.   Our experiments show that automatically acquired disjointness axioms can improve the quality of common ontology mappings (cf. Section 3.4).

3.2 LeDA

Our approach to the automatic acquisition of disjointness axioms relies on a machine learning classifier that determines disjointness of any two classes. The classifier is trained based on a "Gold Standard" of manually created disjointness axioms, i.e. pairs of classes each of which is associated with a label – "disjoint" or "not disjoint" – and a vector of feature values. As in our earlier experiments [VVSH07], we used a variety of lexical and logical features, which we believe to provide a solid basis for learning disjointness. These features are used to build an overall classification model on whose basis the classifier can predict disjointness for previously unseen pairs of classes.

We implemented all features and auxiliary methods for training and classification within the open-source framework LeDA[1] (*Learning Disjointness Axioms*), a complete redesign and re-implementation of our original prototype. LeDA is open-source and publicly available under the LGPL license.

3.3 Features for Learning Disjointness

In the following, we give a brief overview of the 14 features we used for the experiments reported in Section 3.4. The current feature set differs from the original one [VVSH07] in that it focuses more on lexical and ontology-based similarity, which turned out to work very well in previous experiments. At the same time, we omitted several "weak" features including, e.g., OntoClean meta-properties and enumerations.

Taxonomic Overlap.   In description logics, two classes are disjoint *iff* their "taxonomic overlap", i.e. the set of common individuals *must* be empty. Because of the open world assumption in OWL, the individuals of a class do not necessarily have to *exist* in the ontology. Hence, the taxonomic overlap of two classes is considered not empty as long as there *could* be common individuals within the domain that is modeled by the ontology. Following these considerations, we developed several methods to compute the actual or possible overlap of two classes. Both of the following formulas are based on the Jaccard similarity coefficient [Jac12].

$$\mathsf{f}_{overlap_i}(c_1, c_2) = \frac{|\{i \in I | c_1(i) \wedge c_2(i)\}|}{|\{i \in I | c_1(i) \vee c_2(i)\}|} \qquad \mathsf{f}_{overlap_c}(c_1, c_2) = \frac{|\{c \in C | c \sqsubseteq c_1 \sqcap c_2\}|}{|\{c \in C | c \sqsubseteq c_1 \sqcup c_2\}|}$$

These two features are complemented by f_{sub}, that represents a particular case of taxonomic overlap, while at the same time capturing negative information such as class complements or already existing disjointness contained in the ontology. The value of f_{sub} for any pair of classes c_1 and c_2 is 1 for $c_1 \sqsubseteq c_2 \vee c_2 \sqsubseteq c_1$, 0 for $c_1 \sqsubseteq \neg c_2$ and $undefined$ otherwise.

$$\mathsf{f}_{sub}(c_1, c_2) = \begin{cases} 1 & c_1 \sqsubseteq c_2 \vee c_2 \sqsubseteq c_1 \\ 0 & c_1 \sqsubseteq \neg c_2 \\ \text{undefined} & \text{otherwise} \end{cases} \qquad (3.1)$$

Note that subsumption as well as taxonomic overlap greater than zero mostly, but not necessarily implies non-disjointness. This particularly holds when the respective feature values are computed based on a learned background ontology (see further below), but also for many of the lightweight ontologies that we target with LeDA.

---

[1] http://ontoware.org/projects/leda/

Semantic Distance. The semantic distance between two classes c_1 and c_2 is the minimum length of a path consisting of subsumption relationships between atomic classes that connects c_1 and c_2 (as defined in [VVSH07]).

$$\mathsf{f}_{path}(c_1, c_2) = \min_{p \in \mathsf{paths}(c_1, c_2)} \mathsf{length}(p) \tag{3.2}$$

Object Properties. This feature encodes the semantic relatedness of two classes, c_1 and c_2, based on the number of object properties they share. More precisely, we divided the number of properties p with $p(c_1, c_2)$ or $p(c_2, c_1)$ by the number of all properties whose domain subsumes c_1 whereas their range subsumes c_2 or vice-versa. This measure can be seen as a variant of the Jaccard similarity coefficient with object properties considered as undirected edges.

Label Similarity. The semantic similarity of two classes is in many cases reflected by their labels – especially, in case their labels share a common prefix or postfix. This is because the right-most constituent of an English noun phrase[2] can be assumed to be the lexical head, that determines the syntactic category and usually indicates the semantic type of the noun phrase. A common prefix, on the other hand, often represents a nominal or attribute adjunct which describes some semantic characteristics of the noun phrase referent. In order to compute the lexical similarity of the two class labels, we therefore used three different similarity measures:

- *Levenshtein*. The Levenshtein distance measures the edit distance of two strings, i.e. it returns the number of insertion, deletion and substitution operations that are required to transform one string into the other.

- *QGrams.* The idea of the QGrams metric is that two strings have a small edit distance if they have many q-grams in common. A q-gram is a substring of the original string with length q. Our implementation of the QGrams feature is based on the *SimMetrics*[3] library, with $q = 3$.

- *Jaro-Winkler.* The Jaro-Winkler distance is a variant of the Jaro distance metric taking into account the number of matching characters, the number of transpositions and the length of a common prefix.

WordNet Similarity. In order to compute the lexical similarity of two classes (their labels, to be precise), we applied two variants of a WordNet-based similarity measure by Patwardhan and Pedersen [PP03][4]. This similarity measure computes the cosine similarity between vector-based representations of the glosses, that are associated with the two synsets.[5] We omitted any sort of word sense disambiguation at this point, assuming that every class label refers to the most frequently used synset it is contained in.

Features based on Learned Ontology. As an additional source of background knowledge about the classes in our input ontology we used an automatically acquired corpus of Wikipedia articles. By querying Wikipedia for each class label[6] we obtained an initial set of articles some of which were disambiguation pages. We followed all content links and applied a simple word sense disambiguation method in order to obtain the most relevant article for each class: For each class label we considered the article to be most

---

[2]At least in English, people seem to prefer noun phrases for labeling classes.

[3]http://www.dcs.shef.ac.uk/~sam/simmetrics.html

[4]http://www.d.umn.edu/~tpederse/similarity.html

[5]In WordNet, a synset is a set of (almost) synonymous words, roughly corresponding to a class or concept in an ontology. A gloss is a textual description of a synset's meaning, that most often also contains usage examples.

[6]Labels that were written as one word, though consisting of nominal compounds or other types of complex noun phrases.

relevant, which had, relative to its length, the highest "terminological overlap" with all of the labels used in the ontology. The resulting corpus of Wikipedia articles was fed into Text2Onto [CV05a] to generate an additional background ontology for each of the original ontologies in our data set (cf. Section 3.4.1), consisting of classes, individuals, subsumption and class membership axioms.

Based on this newly acquired background knowledge, we defined the following features: *subsumption*, *taxonomic overlap of subclasses* and *individuals* – all of these are defined as their counterparts described above – as well as document-based *lexical context similiarity*, which we computed by comparing the Wikipedia article associated with the two classes. This type of similarity is in line with Harris' distributional hypothesis [Har54] claiming that two words are semantically similar to the extent to which they share syntactic contexts.

Note that in order to enable the computation of feature values on the background ontology (e.g. taxonomic overlap of two classes), we had to map each class in the original ontology, i.e. the one to be enriched with disjointness axioms, to its counterpart in the automatically generated background ontology. We did this by determining for each class the one with the most similar label according to the Jaro-Winkler similarity measure.

3.4 Evaluation

In this section, we describe in detail the evaluation of LeDA and our approach to the automatic acquisition of disjointness axioms. As described in Section 3.4.2, we first compared the automatically generated disjointness to a gold standard of manually acquired axioms. In a second step (see Section 3.4.3), we then investigated the usefulness of our approach with regards to the concrete application of ontology alignment.

3.4.1 Scenario

For the evaluation of our approach we used 6 ontologies from the OntoFarm [SVB$^+$05] data set.[7] Since all of them represent knowledge about the same domain, namely the one of scientific conferences and workshops, we hoped that training performed on any of these ontologies would enable us to classify all other ontologies in the data set. Although our approach is domain-independent in principle, it most probably works best if the ontologies used for training and testing share lexical and structural characteristics, which is less likely to be the case for ontologies from completely different domains.

Gold standard.   In order to obtain a reference set of disjointness axioms for training and evaluating LeDA as well as to get an upper bound for the evaluation of mapping debugging, we manually added a minimal and complete number of disjointness axioms to the ontologies described above.[8] For these sets of explicit disjointness axioms, we computed the transitive closure by "materializing" all implicit disjointness relationships (positive examples). All pairs of classes whose disjointness could not be inferred from the initial, minimal set of axioms were considered not disjoint, thus serving as negative examples in the gold standard. This way we obtained a logically "cleaner" and much bigger data set than in of our earlier experiments with learning disjointness.

Reference mappings.   For the evaluation of our approach to mapping debugging we needed a set of reference mappings consisting of equivalence correspondences for all pairs of ontologies. In order to create such reference mappings of sufficiently high quality, three people familiar with the domain of conferences individually constructed mappings for each pair of ontologies. In case of a disagreement the correctness of a correspondence was decided by a majority vote. It turned out that there was only little disagreement with respect to the correctness of correspondences.

---

[7]The ontologies can be obtained from `http://nb.vse.cz/~svabo/oaei2006/`.
[8]A set of disjointness axioms D is minimal with respect to ontology iff for all $d \in D$ we have $O \cup D \setminus \{d\} \not\models d$.

Ontology	Classes	Properties	Disjointness	Added Axioms
CMT	30	59	27	8
CRS	14	17	12	0
CONFTOOL	39	36	43	1
EKAW	77	33	83	25
PCS	24	38	0	23
SIGKDD	51	28	0	64

Table 3.1: Evaluation data sets. The last column shows the number of (explicit) disjointness axioms that were added in the creation of the gold standard.

3.4.2 Learning Disjointness

In this section, we report on the evaluation of our approach to learning disjointness based on a gold standard of manually created axioms (cf. Section 3.4.1). After giving a brief overview of the evaluation settings and relevant baselines, we summarize the main evaluation results. Finally, we conclude with a qualitative analysis of the features described in Section 3.3

Settings

Training and Test Data.   Unlike in our earlier experiments where a single ontology had to serve as a basis for both training and testing, the conference ontologies data set allows us to use $6 \times 5 = 30$ different combinations of ontologies for the evaluation of learning disjointness: for each of the 6 ontologies, we thus performed 5 experiments using each of the remaining ontologies as training data, and finally averaged over the individual results. Note that we removed all previously existing disjointness axioms from the ontologies prior to training and classification, because we wanted to get comparable results for all ontologies, independently of their respective numbers of existing disjointness axioms.

When testing on any of the ontologies, we always classified (and evaluated against) all possible pairs of classes – not just those explicitly marked as disjoint by the user. This is because we hoped that the resulting redundancy would help to rule out incorrectly classified pairs of classes in a post-processing (debugging) step. As a classifier for all experiments, we used Weka's implementation of NaiveBayes with default parameters[9], which turned out to perform slightly better in our initial tests than Decision Trees and SVMs – especially on the smaller data sets.

Baseline and Evaluation Measures.   We generated macro-average values for precision, recall and F-measure[10] by averaging over the respective results on the sets of positive and negative examples. As a reasonable baseline for our evaluation, we computed a majority baseline for accuracy (Acc_{base}), that is defined as the number of examples in the majority class (e.g. "not disjoint") divided by the overall number of examples. The majority baseline represents the performance level that would be achieved by a naïve classifier that labels all entities in the test set with the majority class, i.e. "disjoint" for all ontologies in our data set. This simple, yet efficient strategy is hard to beat, especially for data sets that are relatively unbalanced and biased towards one of the target classes.

Debugging of Learned Disjointness.   In order to resolve logical incoherences introduced by the automatically generated disjointness axioms, we developed a particular debugging approach. Given a set of learned disjointness axioms D making statements about the classes in ontology O, we apply the following procedure: We sort the set of learned disjointness axioms D according to their associated confidence values

---

[9]http://www.cs.waikato.ac.nz/ml/weka/
[10]In the following we use the term F-measure to refer to the F_1-measure, where recall and precision are evenly weighted.

(i.e. probabilities generated by our machine learning classifier) and start with the highest ranked axioms. Each disjointness axiom $d \in D$ is temporarily added to O. Afterwards, the class hierarchy of ontology O is recomputed. If O is still coherent d can be accepted and becomes a permanent part of O. To reduce the amount of reasoning, we additionally check whether $O \models d'$ for all $d' \in D$ that have not been accepted or rejected yet and accept d' in case of entailment. If O is no longer coherent d is rejected and removed from O. This procedure is an efficient approach that both ensures coherence and increases the quality of learning disjointness.

Results

The evaluation shows that our approach can reliable determine disjointness for any given pair of classes. As detailed by Table 3.2, the comparison of our results with the gold standard set of disjointness axioms yields a macro-average F-measure of up to 80.0%. The accuracy (i.e. the fraction of correctly classified pairs) is even higher ranging from 76.5% to 91.3% which is always above the majority baseline Acc_{base}.

	P	R	F	Acc	Acc_{base}	Acc_{debug}
CMT	0.838	0.785	**0.800**	0.841	0.685	0.842
CONFTOOL	0.792	0.765	0.770	0.855	0.803	0.857
CRS	0.809	**0.801**	0.793	0.867	0.824	N/A
EKAW	**0.882**	0.760	0.797	**0.913**	**0.851**	**0.922**
PCS	0.828	0.749	0.767	0.812	0.663	N/A
SIGKDD	0.778	0.641	0.679	0.765	0.704	0.767

Table 3.2: Results of learning disjointness averaged over all training ontologies (NaiveBayes classifier; macro-avg. precision, recall and f-measure)

Debugging the sets of automatically acquired disjointness axioms as described in Section 3.4.2 further improves the average accuracy, e.g., from 91.3% to 92.2% Acc_{debug} for EKAW. Table 3.3 shows a more detailed evaluation of this post-processing step for individual pairs of training and test ontologies. The first column lists the ontologies that were enriched with automatically acquired disjointness axioms, whereas the second column indicates the respective training ontologies, i.e. the ontologies that were used to setup the classifier. For each training ontology, we obtained an incoherent variant of the original ontology and an accuracy value Acc that was computed by comparison with the respective gold standard. The last column of Table 3.3 lists the relative numbers of disjointness axioms that had to be removed in order to debug the ontology. For example, 2.35% of the learned disjointness axioms had to be removed from EKAW after training had been performed on the CRS ontology. In all cases, debugging further improved the overall quality of the learning results, which led to an increased accuracy (Acc_{debug}).

The complete data sets of our learning disjointness experiments – including the gold standard as well as all the training data and classification results – are available online and can be downloaded from the LeDA homepage.[11]

Feature Ranking

Table 3.4 shows a ranking of our features (see Section 3.3) with respect to their performance on our data set. The first column contains the gain ratio [Qui93] values that were computed by averaging over all training data sets.

Not surprisingly, $f_{overlap_c}$ and f_{sub} performed best on our data set. This is because we exploited the taxonomy in the creation of the gold standard, assuming the ontologies to be carefully designed. A different

---

[11]http://ontoware.org/projects/leda/

	Training on	Acc	Acc_{base}	Acc_{debug}	Removed
CMT	CRS	0.830	0.685	0.839	1.14%
CONFTOOL	CRS	0.869	0.803	0.881	1.41%
EKAW	CMT	0.925	0.851	0.932	0.76%
	CRS	0.892		0.913	2.35%
	PCS	0.919		0.940	2.25%
	SIGKDD	0.904		0.911	0.89%
SIGKDD	CMT	0.779	0.704	0.780	0.09%
	CONFTOOL	0.776		0.776	0.09%
	CRS	0.773		0.781	0.90%
	PCS	0.744		0.744	0.11%

Table 3.3: Individual results of learning disjointness after debugging (NaiveBayes classifier)

methodology for acquiring the reference set of disjointness axioms and taxonomies of lower quality as in our earlier experiments would probably have led to different results.

Two features, $f_{overlap_i}$ and f_{doc} did not contribute at all, which is easy to explain for the first one, because there are no individuals contained in any of the ontologies, but not completely obvious for the f_{doc}, i.e. the document-based lexical context similarity. We assume that the performance of this feature suffers from the fact that only very few classes had associated Wikipedia articles in our experiments.

Gain Ratio	Feature	Description
0.53953417	$f_{overlap_c}$	Taxonomic overlap wrt. subclasses
0.52329850	f_{sub}	Subsumption
0.10352250	f_{prop}	Object properties
0.06150700	$f_{overlap_c}^b$	Taxonomic overlap wrt. subclasses (learned ontology)
0.04459383	f_{sub}^b	Subsumption (learned ontology)
0.03102233	f_{qgrams}	Label similarity (QGrams)
0.02297483	f_{wn_1}	WordNet similarity (Patwardhan-Petersen v1)
0.02070767	$f_{jaro-winkler}$	Label similarity (JaroWinkler)
0.01720867	$f_{levenshtein}$	Label similarity (Levenshtein)
0.00706467	f_{path}	Semantic distance
0.00089500	f_{wn_2}	WordNet similarity (Patwardhan-Petersen v2)
0.00010067	$f_{overlap_i}^b$	Taxonomic overlap wrt. instances (learned ontology)
0.0	f_{doc}	Lexical context similarity (Wikipedia articles)
0.0	$f_{overlap_i}$	Taxonomic overlap wrt. instances

Table 3.4: Feature ranking: average gain ration (full training set)

3.4.3 Ontology Alignment

After we had shown the principle feasibility of learning disjointness (cf. Section 3.4.2), we investigated the usefulness of our approach within a concrete application scenario. In this section, we report on a set of recent experiments [MVS08], that aim to show the benefit of learned disjointness axioms for debugging ontology mappings.

Debugging Mappings

Logical inconsistencies (or incoherences) are an important indicator of potential modeling errors – including errors, that have been introduced by imperfect results of ontology alignment approaches. However, lightweight ontologies usually lack the logical expressivity, e.g., in the form of class complement constructors or cardinality restrictions, which is required to provoke such logical inconsistencies. Merging these ontologies with the help of simple correspondences such as equivalence or subsumption most often yields a perfectly consistent and coherent ontology – with no logical contradictions, that might indicate incorrect mapping axioms.

Our approach to mapping debugging is based on the hypothesis that an automatic enrichment of lightweight ontologies can facilitate the process of mapping revision. By adding automatically (or manually) generated disjointness axioms to merged ontologies we hope to increase their expressivity so that incorrect mappings become evident as logical inconsistencies. Algorithm 1 shows the core of our approach to mapping debugging. First, all mapping axioms $c \in M$ are sorted according to their confidence values. In a second step, starting with the highest ranked axiom, the algorithm iterates over all mappings and for each $c \in M$ removes all $c' \in M$ that form a minimal conflict set with c.

Algorithm 1

RESOLVECONFLICTS($\mathcal{M}$)

1:   $\mathcal{M}' \leftarrow \emptyset$
2:   SORTDESCENDING($\mathcal{M}$)
3:   **while** $\mathcal{M} \neq \emptyset$ **do**
4:      $c \leftarrow$ REMOVEFIRSTELEMENT($\mathcal{M}$)
5:      $\mathcal{M}' \leftarrow \mathcal{M}' \cup \{c\}$
6:      **for all** $c' \in \mathcal{M}$ **do**
7:         **if** ISCONFLICTPAIR(c, c') **then**
8:            REMOVEELEMENT($\mathcal{M}, c'$)
9:         **end if**
10:     **end for**
11: **end while**
12: **return** $\mathcal{M}'$

Results

Table 3.5 summarizes the results of our evaluation experiments. The second column ("generated mappings") presents recall (R), precision (F) and F-measure (F) of the original mappings, generated by state-of-the-art ontology alignment tools. As shown by the last column, these values can be improved significantly by using learned disjointness axioms for mapping debugging. However, the impact of the automatically acquired axioms is still lower than the one of our "reference disjointness", i.e. the manually created disjointness axioms of the gold standard.

Matching System	Automatically generated mappings				Debugged mappings based on							
					reference disjointness				learned disjointness			
	#	P	R	F	#	P	R	F	#	P	R	F
FalconAO-07	123	0.813	0.787	0.800	115	0.852	0.772	0.810	110.40	0.855	0.743	0.795
HMatch-06	203	0.394	0.630	0.485	113	0.690	0.614	0.650	100.04	0.720	0.567	0.634
RiMOM-06	332	0.298	0.780	0.431	233	0.425	0.780	0.550	210.64	0.457	0.757	0.570
OLA-07	389	0.254	0.780	0.383	233	0.421	0.772	0.545	204.80	0.431	0.694	0.532

Table 3.5: Number of correspondences (#), precision (P), recall (R) and F-measure (F) aggregated over input mappings.

3.5 NeOn Toolkit Plugin

In this section, we present a tightly-coupled GUI plugin that serves as a graphical frontend for LeDA. By supporting both the training as well as the classification phase of our approach, this plugin facilitates a customized generation of disjointness axioms for various domains.

3.5.1 User Guide

LeDA view.   The main view of LeDA is depicted by Figure 3.1. A training or classification process can be triggered by means of a context menu, that is accessible by clicking on an OWL ontology in the navigator view on the left (*Learn Disjointness*), or by selecting *Run LeDA* from the main *LeDA* menu.
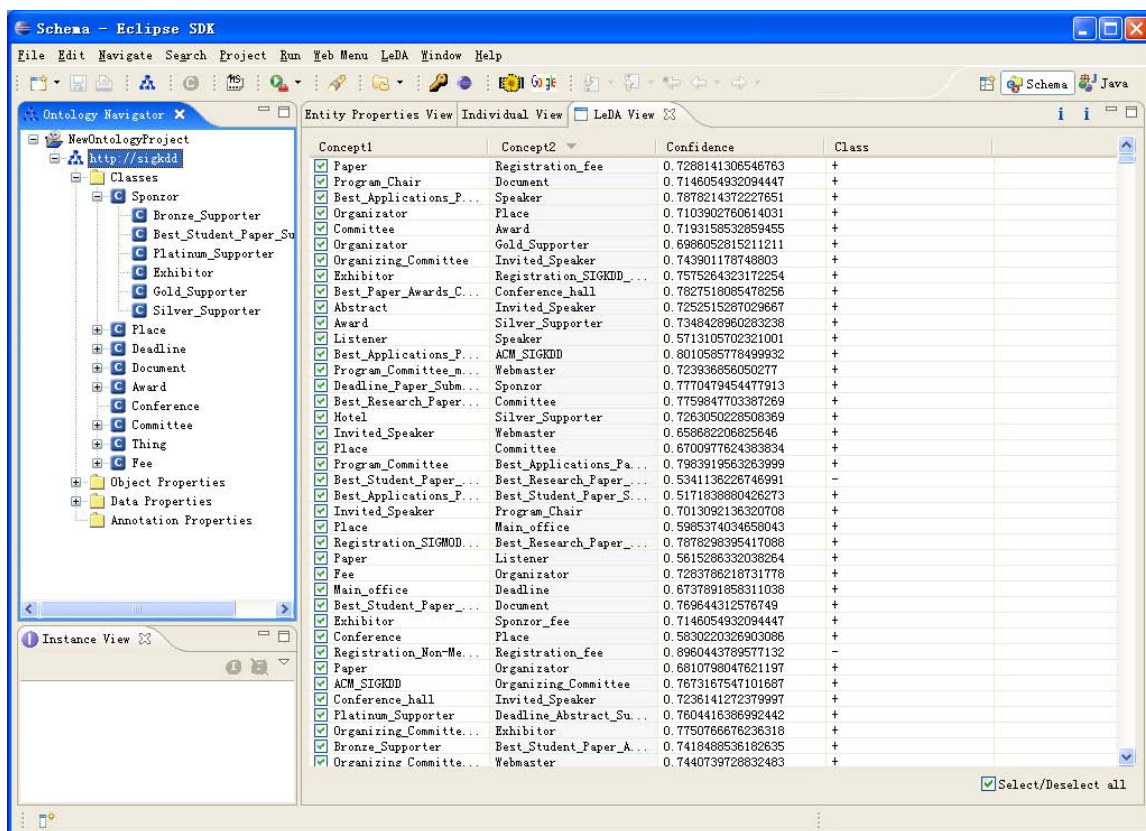

Figure 3.1: LeDA plugin: presentation of results

Preferences.   The *preference page* is accessible from the main menu of Eclipse (*Window → Preferences... → LeDA Preferences*). As shown by Figure 3.2, it allows for setting a variety of parameters for both training and classification:

- **Training or classification**

- **Classifier:** The Weka [12] classifier to be used by LeDA.

- Training

    - **Training ARFF:** This ARFF file will be generated automatically at the end of the training phase. At this point, it will contain all the necessary information for creating a classification model.

    - **Training file:** This is an optional, but recommended parameter. If no training file is specified, positive and negative examples will need to be generated from the training ontology – a process, that is very time-consuming.

    - **Training ontology:** This has to be an ontology, which contains a *complete* set of manually created disjointness axioms.

- Classification

    - **Training ARFF:** The training ARFF file must have been generated in a preceding training phase.

    - **Classification ARFF:** This file will be generated automatically as soon as the classification phase is finished and mainly serves debugging purposes.

    - **Input ontology:** This parameter specifies the ontology, which is to be enriched by disjointness axioms.

3.5.2   Installation

The installation of this plugin does not require specific configuration steps, that might demand for in depth explanations. It can be performed by using the standard plugin update mechanisms of the NeOn Toolkit.
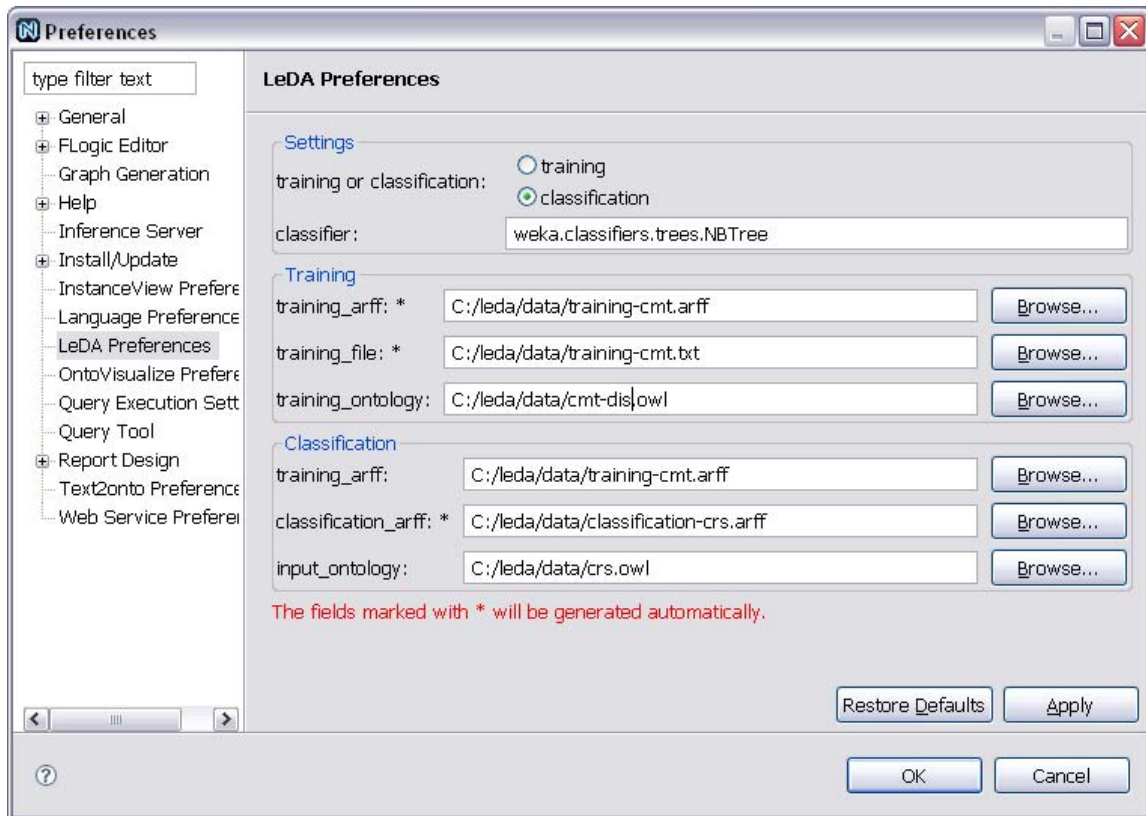
---

[12]`http://www.cs.waikato.ac.nz/~ml/weka/`

Figure 3.2: LeDA plugin: configuration

3.6   Conclusion

In this chapter, we have presented an approach to the automatic acquisition of disjointness axioms. Our detailed evaluation shows that we can learn disjointness with an accuracy that is sufficient for an application such as ontology alignment. The implementation of our approach, the open-source framework LeDA, is available as a plugin for the NeOn Toolkit.

Chapter 4

Learning Class Descriptions

A major shortcoming of lightweight ontologies is their lack of expressivity when it comes to defining the intended meaning of classes. In order to overcome this shortcoming and to facilitate a fine-grained axiomatization of atomic classes, we developed a semi-automatic approach to the acquisition of complex class descriptions. As detailed in the following, we embedded our approach into a conceptual framework of ontology refinement by methodic expert interrogation. The implementation of our framework provides a maximum of user guidance, while at the same time minimizing the manual effort by combining lexical knowledge acquisition with relational exploration.

In Section 4.1, we discuss a few aspects of context sensitivity, that we consider to be relevant within this chapter. Section 4.2 motivates our approach to the acquisition and refinement of complex class descriptions. Theoretical and technical details are given in Section 4.3 and 4.4, followed by an illustrative example in Section 4.5. In Section 4.6, we conclude with a brief summary.

4.1   Context sensitivity for networked ontologies

The automatic acquisition and refinement of complex class descriptions is related to our understanding of context as a semantic modifier in the following ways:

Conceptual views.   Depending on their respective origin, different natural language definitions can represent different views on a particular class. In this sense, the automatic generation of class descriptions from definitions is an effective means to formalize distinct interpretations of classes and to determine differences between people's understanding of ontology entities.

Uncertainty values.   Our approach to the automatic acquisition of class descriptions is capable of measuring uncertainty in various ways. First, the reliability of definitory resources can have an impact on the trustworthiness that is attributed to the resulting description. Second, the generation of multiple related class descriptions from one or more resources facilitates the computation of what could be called "axiomatic support". Axiomatic support can be defined as the relative number of times a particular axiom was generated by an ontology learning component. Since the methods which are applied by such a component can generate axioms of different complexity, it is often necessary to define the axiomatic support based on some normal form of the axioms. Although the current version of RELExO does not handle uncertainty values, that are generated by LExO, i.e. the initial acquisition of class descriptions, taking those into account is mainly a problem of the underlying ontology reasoner.

Ontology alignment.   Some ongoing experiments aim to show the benefit of more expressive class descriptions for the task of ontology alignment. Furthermore, the iterated application of our approach often yields partially overlapping class descriptions, which can be represented as complex mapping axioms.

4.2 RELExO

The more complex an ontology or the bigger a knowledge base is, the more difficult is its extension, evaluation and refinement. In practical scenarios with medium to large size ontologies, it is almost impossible even for an experienced ontology engineer to anticipate all the logical consequences of a modeling activity such as the addition of a class or axiom. At the same time, domain experts who are presented possible formalizations of their knowledge often cannot tell if a given set of axioms sufficiently approximates their conceptualization. In this chapter, we therefore propose a systematic, reasoner-aided approach to ontology acquisition and refinement. It combines an approach to learning expressive class descriptions from textual definitions with relational exploration (RE), a technique based on the well-known attribute exploration algorithm from formal concept analysis (FCA).

By asking decisive questions to clarify still undefined parts of the knowledge base, the process of relational exploration provides user guidance and forces important modeling decisions. Another advantage of relational exploration is that the obtained results are logically crisp and naturally consistent. Moreover, the acquired information is complete with respect to certain well-defined logic fragments of OWL DL. Yet, one major short-coming of RE is that due to the aimed-at completeness, the number of asked questions grows rapidly with the number of involved concepts and roles. We therefore propose a tight integration of RE with lexical ontology learning techniques to focus the exploration process on domain-relevant classes and to automatically answer some of the questions, thus reducing runtime and workload for the expert. In addition, the implementation of our approach relies on an OWL DL reasoner to determine whether the answer to a question posed by the exploration algorithm can be deduced from a previously given background knowledge ontology.

In the following, we show that integrating the two directions of knowledge acquisition, ontology learning and relational exploration, helps to overcome the disadvantages of either approach and to increase the effectiveness and efficiency of ontology refinement. The framework proposed as part of this thesis is to the best of our knowledge the first approach to combining techniques for ontology learning and attribute exploration in a synergetic way.

4.3 Acquisition of Class Descriptions

4.3.1 Implementation: LExO

LExO[1] (Learning EXpressive Ontologies) [VHC07] is an approach towards the automatic generation of ontologies featuring the expressiveness of OWL DL. The core of LExO is a syntactic transformation of definitory natural language sentences into description logic axioms. Given a natural language definition of a class, LExO starts by analyzing the syntactic structure of the input sentence. The resulting dependency tree is then transformed into a set of OWL axioms by means of manually engineered transformation rules. Possible input resources for LExO include all kinds of definitory sentences, i.e. universal statements about concepts, that can be found in online glossaries such as Wikipedia[2], comments in the ontology, or simply given by a domain expert.

In the following, we provide a step-by-step example to illustrate the complete transformation process. For more (and more complicated) examples please refer to Section 4.3.4. We assume that we would like to refine the description of the class ENZYME which could be part of a bioinformatics ontology:

Enzymes are proteins that catalyze chemical reactions.

Initially, LExO applies the Minipar dependency parser [Lin98] in order to produce a structured output as shown in Figures 4.1 and 4.2. Every node in the dependency tree contains information about the token such as its lemma (base form), its syntactic category (e.g. *N* (noun)) and grammatical role (e.g. *subj*), as

---

[1] http://ontoware.org/projects/lexo/
[2] http://en.wikipedia.org

```
E1 ( () fin C * )
1  ( Enzymes enzyme N 2 s (gov be) )
2  ( are be VBE E1 i (gov fin) )
3  ( proteins protein N 2 pred (gov be) )
E3 ( () enzyme N 3 subj (gov protein) (antecedent 1) )
E0 ( () fin C 3 rel (gov protein) )
4  ( that ~ THAT E0 whn (gov fin) (antecedent 3) )
5  ( catalyze ~ V E0 i (gov fin) )
E4 ( () that THAT 5 subj (gov catalyze)  (antecedent 3) )
6  ( chemical ~ N 7 nn (gov reaction) )
7  ( reactions reaction N 5 obj (gov catalyze) )
```

Figure 4.1: Minipar Dependency Tree

well as its surface position. Indentation in this notation visualizes direct dependency, i.e. each child node is syntactically dominated by its parent.
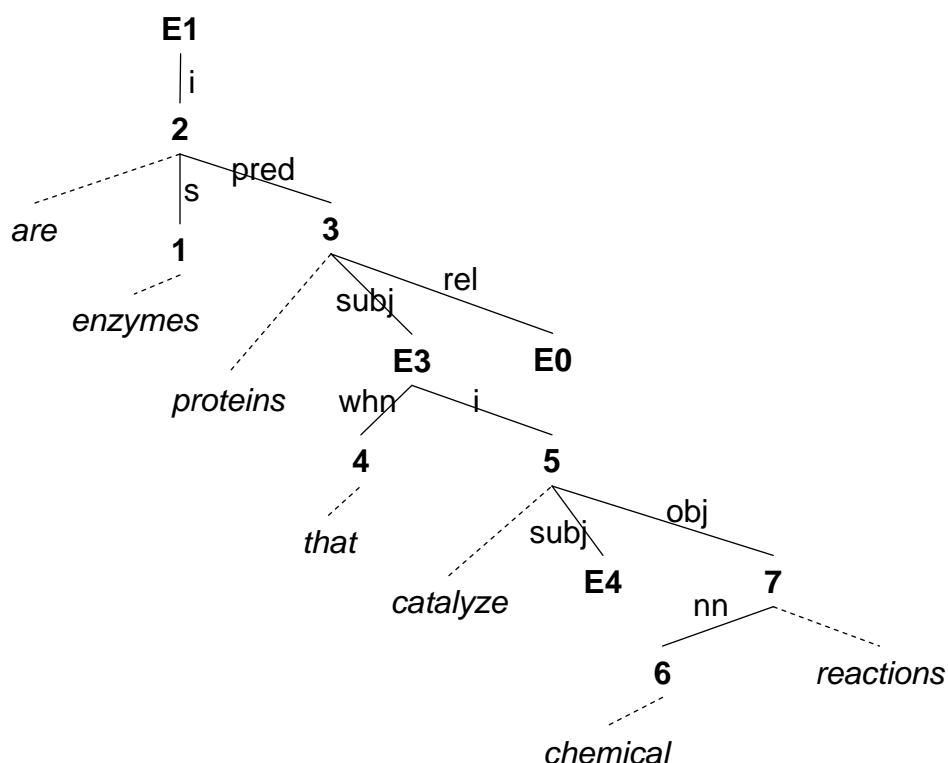

Figure 4.2: Visualization of Dependency Tree

This dependency structure is now transformed into an XML-based format (see Figure 4.3) in order to facilitate the subsequent transformation process, and to make LExO more independent of the particular parsing component.

The set of rules which are then applied to the XML-based parse tree make use of XPath expressions for transforming the dependency structure into one or more OWL DL axioms. Figure 4.4 shows a few examples of such transformation rules in original syntax. Each of them consists of several arguments (e.g. *arg_1:...*), whose values are defined by an optional prefix, i.e. a reference to a previously matched argument (*arg_0*), plus an XPath expression such as */C[@role='rel']* being evaluated relative to that prefix. The last lines of each transformation rule define one or more templates for OWL axioms, with variables to be replaced by the values of the arguments. Complex expressions such as *0-1* allow for "subtracting" individual subtrees from

```
<?xml version="1.0" encoding="UTF-8"?> <root>
 <C id="E1" pos="0">
  <VBE id="2" pos="2" role="i" phrase="are" base="be">
   <N id="1" pos="1" role="s" phrase="enzymes" base="enzyme"/>
   <N id="3" pos="3" role="pred" phrase="proteins" base="protein">
    <N id="E3" pos="4" role="subj" base="enzyme"/>
    <C id="E0" pos="5" role="rel">
     <THAT id="4" pos="6" role="whn" phrase="that" antecedent="3"/>
     <V id="5" pos="7" role="i" phrase="catalyze">
      <THAT id="E4" pos="8" role="subj" base="that" antecedent="3"/>
      <N id="7" pos="10" role="obj" phrase="reactions" base="reaction">
       <Det id="6" pos="9" role="nn" phrase="chemical"/>
      </N>
     </V>
    </C>
   </N>
  </VBE>
 </C>
</root>
```

Figure 4.3: XML Representation of Dependency Tree

the overall tree structure. A more complete listing of the transformation rules we applied can be found further below.

```
rule: relative clause {
    arg_0:   //N
    arg_1:   arg_0 /C[@role='rel']
    arg_2:   arg_1 /V
    result: [equivalent 0 [and 0-1 2]]
} rule: verb and object {
    arg_0:   //V
    arg_1:   arg_0 /N[@role='obj']
    result: [equivalent 0 [some 0-1 1]]
    result: [subObjectPropertyOf 0 0-1]
}
```

Figure 4.4: Transformation Rules

A minimal set of rules for building a complete axiomatization of the ENZYME example could be, e.g., *Copula*, *Relative Clause*, *Transitive Verb Phrase* and *Intersective Adjective* (see Section 4.3.2). The resulting list of axioms (see Figure 4.5) in KAON2[3] internal syntax is directly fed into the ontology management system which interprets the textual representation of these axioms, and finally builds an unfolded[4] class description as shown in Figure 4.6.

```
[equivalent lexo:enzymes
    lexo:proteins_that_catalyze_chemical_reactions]
[equivalent lexo:proteins_that_catalyze_chemical_reactions
    [and lexo:proteins lexo:catalyze_chemical_reactions]]
[equivalent lexo:catalyze_chemical_reactions
    [some lexo:catalyze lexo:chemical_reactions]]
[equivalent lexo:chemical_reactions
    [and lexo:chemical lexo:reactions]]
```

Figure 4.5: Resulting Axioms

Obviously, all parts of this class description have to be normalized. After the normalization, the final, axiomatization in DL syntax reads:

ENZYME $\equiv$ PROTEIN $\sqcap$ $\exists$catalyze.(CHEMICAL $\sqcap$ REACTION)

---

[3]http://kaon2.semanticweb.org

[4]By *unfolding*, a term borrowed from logic programming, we mean transformations like that of $\{A \equiv \exists R.B, C \equiv A \sqcap D\}$ to $\{C \equiv \exists R.B \sqcap D\}$. The specific for of output which we receive allows us to remove many of the newly generated class names by unfolding, in order to obtain a more concise output.

```
[ equivalent lexo:enzymes
    [and lexo:proteins [some lexo:catalyze [and lexo:chemical lexo:reactions ]]]]
```

Figure 4.6: Class Description (unfolded)

Additionally, it may be necessary to map the ontology elements of the axiomatization to already existing content of the ontology before the results can be used to generate suggestions for ontology changes. As shown by the large body of research done in the domain of ontology mapping, this task is not trivial at all. Semantic ambiguities of labels (e.g. *homonymy* or *polysemy*), as well as the fact that a single entity or axiom in the ontology can have arbitrarily many lexicalizations – differing even in their syntactic category – make it necessary to consider a multitude of possible mappings. Moreover, idiomatic expressions, i.e. expressions whose meaning cannot be directly derived from the meaning of their individual components, need to be treated properly. Therefore, in addition to integrating a state-of-the-art mapping framework, a significant degree of user involvement will be unavoidable in the end.

4.3.2 Transformation Rules

Table 4.1 gives an overview of the most frequently used transformation rules. Each row in the table contains the rule name (e.g. *Verb with Prepositional Complement*) and an expression describing the natural language syntax matched by that rule – like, for example, $V_0 \, Prep_0 \, NP(\textit{pcomp-n})_0$, where V_0 represents a verb, $Prep_0$ a preposition and $NP(\textit{pcomp-n})$ denotes a noun phrase acting as a prepositional complement. Please note that these expressions are very much simplified for the sake of presentation. The last column shows the OWL axioms generated in each case, where X denotes the atomic class name represented by the surface string of the complete expression matched by the regarding transformation rule.

It is important to emphasize that this set of rules is by no means exhaustive, nor does it define the only possible way to perform the transformation. In fact, there are many different modeling possibilities, and the choice and shape of the rules very much depends on the underlying application, the domain of interest or individual modeling preferences of the user (see example TETRAPLOID in Section 4.3.4).

Rule	Natural Language Syntax	OWL Axioms
Disjunction	NP_0 `or` NP_1	$X \equiv NP_0 \sqcup NP_1$
Conjunction	NP_0 `and` NP_1	$X \equiv NP_0 \sqcap NP_1$
Determiner	$Det_0 \, NP_0$	$X \equiv NP_0$
Intersective Adjective	$Adj_0 \, NP_0$	$X \equiv Adj_0 \sqcap NP_0$
Subsective Adjective	$Adj_0 \, NP_0$	$X \sqsubseteq NP_0$
Privative Adjective	$Adj_0 \, NP_0$	$X \sqsubseteq \neg NP_0$
Copula	$NP_0 \, VBE \, NP_1$	$NP_0 \equiv NP_1$
Relative Clause	$NP_0 \, C(\textit{rel}) \, VP_0$	$X \equiv NP_0 \sqcap VP_0$
Number Restriction	$V_0 \, Num \, NP(\textit{obj})_0$	$X \equiv =Num \, V_0.NP_0$
Negation (not)	`not` $V_0 \, NP_0$	$X \sqsubseteq \neg\exists V_0.NP_0$
Negation (without)	NP_0 `without` $NP(\textit{pcomp-n})_1$	$X \equiv NP_0 \sqcap \neg with.NP_1$
Participle	$NP_0 \, VP(\textit{vrel})_0$	$X \equiv NP_0 \sqcap VP_0$
Transitive Verb Phrase	$V_0 \, NP(\textit{obj})_0$	$X \equiv \exists V_0.NP_0$
Verb with Prep. Compl.	$V_0 \, Prep_0 \, NP(\textit{pcomp-n})_0$	$X \equiv \exists V_0\_Prep_0.NP_0$
Noun with Prep. Compl.	$NP_0 \, Prep_0 \, NP(\textit{pcomp-n})_1$	$X \equiv NP_0 \sqcap \exists NP_0\_Prep_0.NP_1$
. . .	. . .	. . .

Table 4.1: Transformation Rules

4.3.3   Technical Discussion

The syntactic transformations proposed in Section 4.3 create a set of OWL axioms which can be used to extend the axiomatization of any given class in an ontology. Our naive implementation of this approach is as simple as efficient, but obviously requires a significant amount of manual or automatic post-processing. This is to a major extent due to a number of problems that relate to limitations of the linguistic analysis and the transformation process. In the following, we will discuss several of these technical issues in more detail, and present possible solutions. For a discussion of more fundamental problems that arise from differences between lexical and ontological semantics please refer to [VHC07].

A significant objection one might have with respect to the technical implementation of our approach certainly refers to the rather sophisticated linguistic analysis which is required prior to the actual transformation process. Indeed, the Minipar dependency parser we use to obtain a syntactic analysis sometimes fails to deliver a parse. This frequently happens in the case of ill-formed or structurally complex sentences, or wrongly resolved *syntactic ambiguities* such as prepositional phrase attachments. However, dependency parsers are known to be much more robust than parsers using phrase structure grammar, for example. And as most of our transformation rules can be mapped to surface structure heuristics (see Table 4.1) in a relatively straightforward way, a chunker or shallow parser could complement Minipar in case of failure. Efficiency is not so much an issue as the parser is extremely fast, processing a few hundred sentences per second.

However, there are more severe problems apart from the quality or efficiency of the syntactic analysis. Many of them concern *semantic ambiguity* related to quantifier scope or homonymy (e.g. "*net*"). Both types of problems are not appropriately handled at the moment. And our implementation also lacks an *anaphora resolution* step which would help to identify antecedents of pronouns (e.g. "*its atmosphere*") or nominal anaphora, for instance. Although some types of coreference including relative pronouns can be handled by Minipar itself, the language we defined for describing the transformation rules is not expressive enough to deal with phenomena such as long distance dependencies or deictic expressions. Therefore, user intervention is still essential during the post-processing phase to replace pronouns and to map co-referring nominals to the same class. Similarly, depending on the desired degree of modeling granularity, user input might be required to support the semantic analysis of *compound nominals* (e.g. "*Pair Trawling*").

Moreover, the different semantics of *adjectives* are not taken into account by the translation rules. Ideally, one would have to distinguish between at least three types of adjectives – *subsective* (Young_Fish ⊑ Fish), *intersective* (Sexual_Maturity ⊑ Sexual ⊓ Maturity) and *privative* (Fake_Fish ⊑ ¬Fish). But since an automatic classification of adjectives into these classes as proposed by [AG06], for example, is a very challenging task, we currently assume intersective semantics for all adjectives. Even more difficult is the semantics of *adverbs* (e.g. "*completely surrounded*") and some types of auxiliary verbs which express a spatial, temporal or behavioral modality (e.g. "*can support life*"). And of course, temporal relationships expressed by past or future *verb tense* are also very difficult to handle without temporal reasoning.

Another problem which is not yet sufficiently handled by our transformation rules are so-called *empty heads*, i.e. nominals which do not contribute to the actual meaning of a genus phrase. In particular, the rules relying on Hearst-style patterns [Hea92] for the identification of hyponymy relationships may be mislead by expressions such as *one*, *any*, *kind*, *type*. This phenomenon has already been described in the literature [GSWB90, CBH85] and could be handled by appropriate exception rules. An alternative solution to this and similar problems could be to increase the expressiveness of the *rule language* used in the transformation process. The language as it is defined by now does not permit the usage of regular expressions, for instance, which might be a valuable means to generalize particular transformation rules. XSLT can help to overcome these limitations and indeed, we have already implemented an alternative back-end for processing the transformation rules.

Finally, our approach is restricted to texts with *definitory character* such as glossary entries or encyclopedic descriptions which have a universal reading and a more or less canonical form, i.e. including a genus category and additional information to distinguish the term from other members of the same category [KPP03].

In order to extend the applicability of LExO to a greater variety of textual resources, one would need a component for the automatic identification of natural language definitions.

4.3.4 Case Study Examples

The FAO Fisheries department has several information and knowledge organization systems to facilitate and secure the long-term, sustainable development and utilization of the world's fisheries and aquaculture. In order to effectively manage the world's shared fish stocks and prevent overfishing, the FAO Fishery systems manage and disseminate statistical data on fishing, data from geographic information systems (GIS), aquaculture information, geographic entities, description of fish stocks, etc. However, even though much of the data is "structured", it is not necessarily represented in a formal way, and some of the information resources are not available through databases but only as parts of websites, or as individual documents or images. Therefore, many or even all of these data sources could be better exploited by bringing together related and relevant information, along with the use of the fishery ontologies, to provide inference-based services for policy makers and national governments to make informed decisions.

A particular application developed within the NeOn project is FSDAS (*Fishery Stock Depletion Alert System*), an ontology-driven decision support system for fisheries managers, assistants to policy makers and researchers. FSDAS is a web-based intelligent agent that uses networked ontologies consisting of various fisheries, taxonomic, and geographical ontologies to aid users in discovering resources and relationships related to stock depletion and to detect probabilities of over-fishing. Fisheries ontologies, which bring together concepts from a number of existing knowledge organization systems, help to improve language-independent extraction and the discovery of information. Their development will allow for managing the complexity of fishery knowledge communities, their meaning negotiation and their deployment by worldwide authorities.

In order to achieve these goals, the ontological model needs to be shaped starting from highly structured FAO information systems, and to develop a learning capacity from this model to incorporate data and information from other less structured systems. Here, ontology learning becomes an integral part of the lifecycle of the fishery ontology. Further, in order for the FSDAS to be effective, it is important that the ontologies and resources it builds on are maintained and kept up-to-date, and that when applying changes to ontologies the consistency of the ontology is guaranteed.

We now illustrate our approach by analysing a number of class descriptions that were automatically generated using the set of rules listed by Table 4.1. The source definitions are were selected from the FAO's fishery glossary[5] with more than 1,000 entries for various fishery-related concepts.

1. ***Data:*** *Facts that result from measurements or observations.*
   $\text{DATA} \equiv \text{FACT} \sqcap \exists \text{result\_from.}(\text{MEASUREMENT} \sqcup \text{OBSERVATION})$

2. ***InternalRateOfReturn:*** *A financial or economic indicator of the net benefits expected from a project or enterprise, expressed as a percentage.*
   $\text{INTERNALRATEOFRETURN} \equiv (\text{FINANCIAL} \sqcup \text{ECONOMIC}) \sqcap \text{INDICATOR} \sqcap \exists \text{indicator\_of.}(\text{NET} \sqcap \text{BENEFIT} \sqcap \exists \text{expected\_from.}(\text{PROJECT} \sqcup \text{ENTERPRISE})) \sqcap \exists \text{expressed\_as.}\text{PERCENTAGE}$

3. ***Vector:*** *An organism which carries or transmits a pathogen.*
   $\text{VECTOR} \equiv \text{ORGANISM} \sqcap (\text{carry} \sqcup \exists \text{transmit.}\text{PATHOGEN})$

4. ***Juvenile:*** *A young fish or animal that has not reached sexual maturity.*
   $\text{JUVENILE} \equiv \text{YOUNG} \sqcap (\text{FISH} \sqcup \text{ANIMAL}) \sqcap \neg \exists \text{reached.}(\text{SEXUAL} \sqcap \text{MATURITY})$

5. ***Tetraploid:*** *Cell or organism having four sets of chromosomes.*
   $\text{TETRAPLOID} \equiv (\text{CELL} \sqcup \text{ORGANISM}) \sqcap =4 \text{ having.}(\text{SET} \sqcap \exists \text{set\_of.}\text{CHROMOSOMES})$

---

[5] http://www.fao.org/fi/glossary/default.asp

6. ***Pair Trawling:*** *Bottom or mid-water trawling by two vessels towing the same net.*
   PAIRTRAWLING ≡
   (BOTTOM ⊔ MIDWATER) ⊓ TRAWLING ⊓ =2 trawling_by.(VESSEL ⊓ ∃tow.(SAME ⊓ NET))

7. ***Sustained Use:*** *Continuing use without severe or permanent deterioration in the resources.*
   SUSTAINEDUSE ≡ CONTINUING ⊓ USE ⊓ ¬∃use_with.((SEVERE ⊔ PERMANENT) ⊓
   DETERIORATION ⊓ ∃deterioration_in.RESOURCES)

8. ***Biosphere****: The portion of Earth and its atmosphere that can support life.*
   BIOSPHERE ≡ PORTION ⊓ ∃portion_of.((EARTH ⊓ ITS ⊓ ATMOSPHERE) ⊓ ∃can_support.LIFE)

Some critical remarks and observations on the examples:

1. This is a simple example, which works out very well.

2. This example shows the complex axiomatizations which can be obtained using our approach. Here (and in other examples) we note that adjectives are so far interpreted as being intersective – we will review this assumption in Section 4.3.3. Another recurring problem is the generic nature of the role of which we tried to solve by designing the transformation rule in a way that it adds a disambiguating prefix to the preposition as a role name (indicator_of). Nevertheless, the output is likely to be a reasonable approximation of the intended meaning and would serve well as suggestion for an ontology engineer within an interactive process of semi-automatic ontology engineering.

3. This is a Minipar parse error. The desired solution would be
   VECTOR ≡ ORGANISM ⊓ (∃CARRY.PATHOGEN ⊔ TRANSMIT.PATHOGEN).

4. Take particular attention to the handling of negation and the present perfect tense.

5. The natural language sentence is actually ambiguous whether the number should be read as *exactly four* or *at least four*, and the role name having is certainly not satisfactory. Even more difficult is how *set of chromosomes* is resolved. A correct treatment is rather intricate, even if modeling is done manually. The class name CHROMOSOMES should probably rather be a nominal containing the class name as individual – which cannot be modeled in OWL DL, but only in OWL Full. Note also that the cardinality restriction is used as a so-called *qualified* one, which is not allowed in OWL DL but is supported by most DL reasoners.

6. *Same* is difficult to resolve. In order to properly model this sentence, one would have to state that two different individuals of the class VESSEL are connected to the same instantiation of NET by means of the tow role. This is *not* expressible in OWL DL as, in the general case, such constructions would lead to undecidability.

7. Apart from the very generic role in and the problem with adjectives already mentioned, this is a complex example which works very well.

8. The possessive pronoun *its* would have to be modeled differently.

4.4 Refinement of Class Descriptions

4.4.1 Relational Exploration

Relational Exploration (introduced in [Rud04] and thoroughly treated in [Rud06]) is a technique based on the well-known attribute exploration algorithm from Formal Concept Analysis (FCA) which is used to interactively clarify underspecified logical dependencies. By forcing particular modeling decisions the exploration

of classes and class extension relationships guarantees completeness with respect to a certain logical fragment, thereby increasing the overall quality of the ontology.

The classical attribute exploration algorithm [Gan99] provides a method for efficiently determining an implicational base of a formal context that is only implicitly known by an expert. The technique of Relational Exploration (RE) extends this algorithm to a DL setting: Given an interpretation $\mathcal{I}$ on a domain Δ and a set M of $\mathcal{SHOIN}$ concept descriptions, the corresponding $\mathcal{I}$-*context* is defined by

$$\mathbb{K}_{\mathcal{I}}(M) := (\Delta, M, I) \text{ with } \delta I C :\Leftrightarrow \delta \in C^{\mathcal{I}}.$$

Then it can be easily shown that implications in $\mathbb{K}_{\mathcal{I}}$ coincide with certain axioms with respect to their validity in $\mathcal{I}$: for $\mathcal{C}, \mathcal{D} \subseteq M$, the implication $\mathcal{C} \rightarrow \mathcal{D}$ holds in $\mathbb{K}_{\mathcal{I}}$ if and only if $\mathcal{I}$ satisfies the DL axiom

$$\bigsqcap \mathcal{C} \sqsubseteq \bigsqcap \mathcal{D}.$$

Hence it is possible to explore DL axioms (more precisely, *general concept inclusion axioms* or short: GCIs) with this techniques. I.e., in an interview-like process, a domain expert has to judge whether a proposed GCI is valid in the domain (formally: the interpretation $\mathcal{I}$) she is describing and in the negative case provide a counterexample. Since OWL DL is based on description logics, the RE method easily carries over to any kind of ontologies specified in that language.

Especially when working in an OWL or DL setting, the open world assumption is omnipresent. Most of the known objects will not be completely specified, i.e. for certain classes it might be unknown whether the considered individual is an instance. Hence, it is essential for exploration methods to be capable of dealing with this kind of information. Lately, there has been significant work on applying FCA results on *partial* information (e.g. described in [Bur91, Gan99]) to the ontology refinement setting. An according approach (briefly sketched in [Rud06]) has been fully theoretically elaborated and implemented as described in [FBS07]. It allows to use partly specified objects as counterexamples for hypothetical implications, hence to handle partial contexts.

4.4.2 Combined Approach

In the sequel, we will describe how LExO and Relational Exploration (RE) can be synergetically combined by giving a comprehensive description of the integrated algorithm. En route, we will briefly mention how other lexical ontology learning techniques could be beneficially used within that process. In addition to the LExO and RE components, an OWL DL reasoner will be applied in order to draw conclusions that are already implicitly present, i.e. entailed by the actual knowledge base making an intervention of the user obsolete.

Creation of new Definitions and Mappings.   We start with an OWL DL ontology $\mathcal{KB}$ to be refined with respect to a (new or already contained) class C, for which a natural language definition is provided by some textual resource. This textual definition is then analyzed by LExO yielding a set $\mathcal{KB}$' of OWL DL axioms as described in Section 4.3. Most likely, some (or even most) of the named classes those axioms refer to will not be present in $\mathcal{KB}$. Therefore, at least the primitive classes amongst those – i.e. those classes not stated to be equivalent to a complex class description[6] – should be linked to $\mathcal{KB}$. There are several ways for doing that. If textual definitions are available, LExO could be employed "recursively", i.e., it might be applied to the definitions of the classes in question in order to obtain other classes that can be linked to $\mathcal{KB}$ more easily. In any case, ontology *mappings* between $\mathcal{KB}$ and $\mathcal{KB}$' could be either added manually or established by one or several of the well-known mapping tools like FOAM[7] [ES05]. So let Map be a (possibly empty) set of respective mapping axioms.

---

[6]which are those occurring explicitly in the normal form from Section 4.3
[7]http://www.aifb.uni-karlsruhe.de/WBS/meh/foam/

Selection of Relevant Classes. In the next step, we stipulate the focus of the subsequent exploration by selecting those atomic classes from $\mathcal{KB} \cup \mathcal{KB}'$ whose logical dependencies are to be clarified. A natural default choice for this would be the set of all named classes from $\mathcal{KB}$', as we might suppose that the (remaining) classes from $\mathcal{KB}$ are modeled in a sufficiently precise way, already – an assumption that might be disproved later on. However, it can be reasonable to include some of the classes from $\mathcal{KB}$ as well. Knowledge extraction methods that determine the relevance of terms (like those offered by Text2Onto [CV05a]) could be employed for an automatic selection or to generate reasonable suggestions. In any case, let $\mathbf{C}$ denote the set of selected attributes.

After this selection of relevant named classes, a basic fact from FCA allows to further restrict $\mathbf{C}$: put into DL notation, it assures the dispensability of a class $C \in \mathbf{C}$ whenever there is a set $\mathbf{D} = \{D_1, \ldots, D_n\} \subseteq \mathbf{C} \setminus \{C\}$ such that $C \equiv D_1 \sqcap \ldots \sqcap D_n$ follows from all knowledge $\mathcal{KB}_\Sigma := \mathcal{KB} \cup \mathcal{KB}' \cup Map$ stated so far.[8] It takes just a little consideration that this is the case iff

$$ \mathcal{KB}_\Sigma \models \bigsqcap \Big\{ D \mid D \in \mathbf{C} \setminus \{C\}, \ \mathcal{KB}_\Sigma \models C \sqsubseteq D \Big\} \sqsubseteq C, $$

such that the elimination of redundant classes from $\mathbf{C}$ requires just $O(|\mathbf{C}|^2)$ reasoner calls in the worst case. Let $\mathbf{C}'$ denote the result of this reduction process.

Exploration. Now we start RE as described in Section 4.4.1 on the concept set $\mathbf{C}'$. A work flow diagram of the procedure is displayed in Figure 4.7. For every hypothetical DL axiom $C_1 \sqcap \ldots \sqcap C_n \sqsubseteq D_1 \sqcap \ldots \sqcap D_m$ brought up by the exploration algorithm:
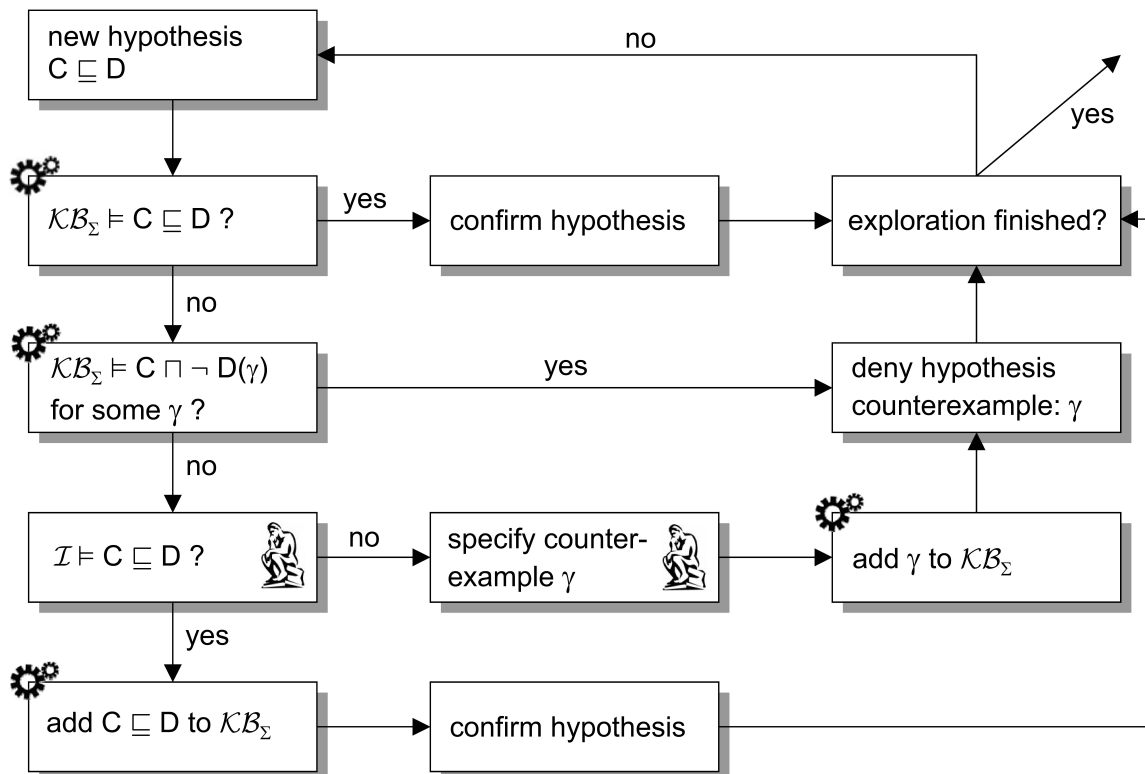

Figure 4.7: Relational Exploration process (the gear wheels indicate ontology management activities including reasoning and updates, whereas the thinker icon marks user involvement).

- Employ the reasoner to check whether this GCI is a consequence of $\mathcal{KB}_\Sigma$. If so, confirm the implication and continue the exploration with the next hypothesis.

---

[8]In FCA terms this can be conceived as a kind of a-priori column-reduction.

- Employ the reasoner to query for all individuals γ with $C_1 \sqcap \ldots \sqcap C_n \sqcap \neg D_i(\gamma)$ for an i from $1, \ldots, m$, i.e., for instances of the class which characterizes the property for being a material counterexample[9] for the hypothetical GCI. Let Γ be the set of individuals retrieved this way. If $\Gamma \neq \emptyset$, select one $\gamma \in \Gamma$ and check for every $C \in \mathbf{C}$ whether $C(\gamma)$ or $\neg C(\gamma)$. Then the counterexample together with the information about the attributes it provably has or has not is passed to the exploration algorithm. Optionally the human expert – possibly assisted by lexical knowledge retrieval tools – might be asked to complete the assertions for γ in order to get a more specific description for it. In any case, after providing γ, the exploration will proceed with the next hypothesis.

- If the DL axiom in question can be neither automatically proved nor declined (the latter meaning $\Gamma = \emptyset$), the human will be asked for the ultimate decision whether the axiom is satisfied in the described domain $\mathcal{I}$ or not. Again, ontology learning tools could support him by suggesting answers endowed with a confidence value, or simply scanning a corpus for potential hints and presenting selected passages.

The exploration terminates after finitely many steps, yet it may also be stopped by the user beforehand. In the latter case, the internal order of the classes from the set $\mathbf{C}'$ is relevant since it determines the order of the posed questions. Hence, it is beneficial to sort those classes wrt. their relevance, possibly based on textual information. After the exploration cycle being finished, we have obtained a refined knowledge base $\mathcal{KB}_\Sigma$ containing the (possibly new) class C endowed with its definition (as extracted from the textual definition) and its interrelationships with concepts from the original knowledge base. Additionally, the "semantic neighborhood" of C has been made logically explicit by interactive exploration. In fact, any subsumption between conjunctions of classes from $\mathbf{C}$ can be decided (i.e. proven or disproven) based on the refined knowledge base. This also shows the advantage of introducing atomic classes for the complex concept descriptions occurring in the LExO output as demonstrated in Section 4.3: although RE as applied in this case[10] deals only with conjunctions on atomic classes, we introduce more expressivity "through the back-door" by having complex definitions for those named classes in our ontological background ready to be exploited by the reasoner.

The synergies provided by the presented combination are manifold: Firstly, the classes contained in the definitions provided by LExO provide a reasonable small to medium size "exploration scope" being crucial for a reasonable application of the RE technique. Secondly, we can use textual information for generating ontological information (a source not accessible to purely logical approaches) yet being able to interactively clarify logical dependencies that have been left open by the text. The latter is done in a guided way ensuring completeness.

Overall, the proposed framework provides means for interactively integrating learned or manually acquired axiomatizations into an existing ontology, while at the same time facilitating their evaluation and refinement.

4.4.3   Implementation: RELExO

In order to prove the feasibility of a synthesis of LExO and Relational Exploration as described in Section 4.4.2, we instantiated our approach by means of a prototypical application named RELExO.[11] RELExO relies upon KAON2[12] as an ontology management back-end and features a simple graphical user interface. Its architecture is depicted by Figure 4.8.

LExO, possibly complemented by other ontology learning components, generates or extends the initial set of axioms $\mathcal{KB}$ (mappings can be added by FOAM, if necessary), and initializes the partial context $\mathbb{K}$ by suggesting a set of attributes $\mathbf{C}$ to the user. The actual refinement process is handled by a RE component

---

[9]Material counterexamples are objects for which is known which part of the conclusion they violate. The exploration algorithm (even the one dealing with partial knowledge) can only make use of this kind of counterexamples.

[10]Actually, RE provides means for exploring GCIs in whole $\mathcal{ALE}$, however we restrict to conjunctions on atomic classes in this example.

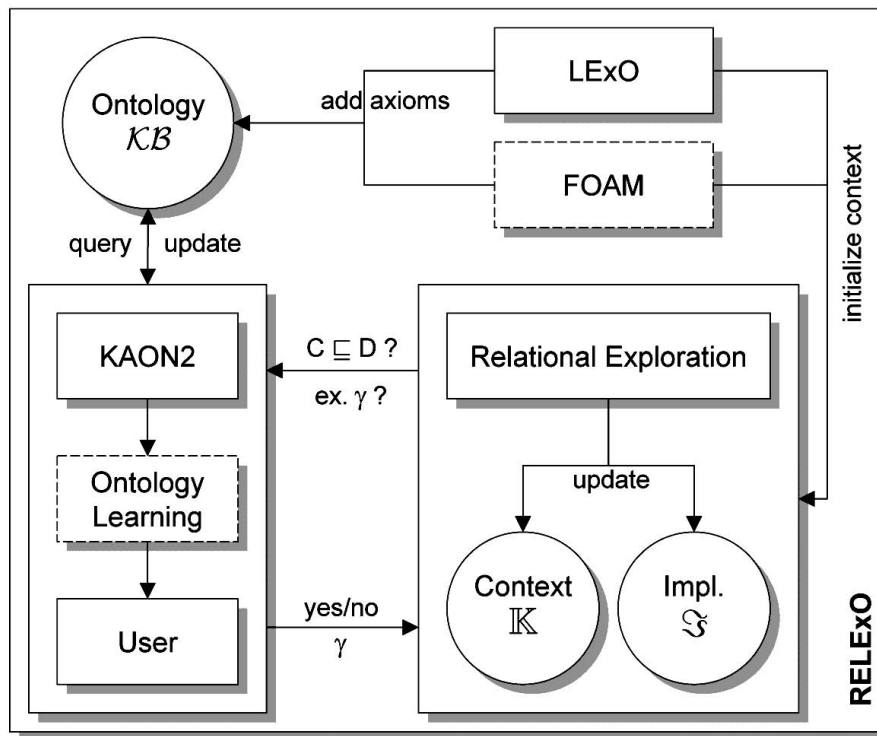[11]Both sources and binaries of RELExO are available for public use and can be downloaded from http://relexo. ontoware.org.

[12]http://kaon2.semanticweb.org

Figure 4.8: RELExO Architecture

which manages the partial context $\mathbb{K}$ and the implication set $\mathfrak{I}$. Both are updated based on answers obtained from the "expert team" constituted by the KAON2 reasoner, an optional ontology learning component as well as the human knowledge engineer.

4.5 Integrated Example

We now illustrate the integrated ontology refinement process which has been elaborated on in Section 4.4.2 by means of a real-world example. The complete material necessary for reproducing this example, i.e. ontologies and screenshots, is contained in the RELExO distribution.

Ontology. The SWRC[13] (*Semantic Web for Research Communities*) ontology is a well-known ontology modeling the domain of Semantic Web research [SBH$^+$05]. Version 0.7 contains 71 classes, e.g., for different types of persons, publication, and events, 48 object properties, 46 datatype properties, and an overall number of 672 axioms. Its expressiveness is slightly beyond OWL DLP [GHVD03] featuring subsumption, properties, and a few disjointness axioms. The ontology serves as a basis for semantic annotation in the AIFB web portal[14] that manages information about more than 2,000 persons, projects, and publications. For the purpose of our experiment, we exported all instance data stored in the AIFB portal into one single OWL file (more than 3 Megabytes in RDF syntax), and merged it with the corresponding TBox, i.e. the latest version of SWRC. After minor syntactic corrections (removing non XML-compliant characters), we obtained a considerably large ontology. Debugging with RaDON[15] revealed two inconsistencies caused by conflicting range specifications of data properties which could be fixed without difficulty.

Subsequently (in order to keep the example simple and rule out a few trivial questions that would otherwise come up in the exploration phase), we added axioms stating the disjointness of the SWRC top-level classes

---

[13]http://ontoware.org/projects/swrc/
[14]http://www.aifb.uni-karlsruhe.de
[15]http://radon.ontoware.org

PERSON, EVENT, and PUBLICATION – obviously true axioms yet not present in the current version of this ontology. Those axioms could also have been generated automatically by techniques for learning disjointness as described in [VVSH07] and Chapter 3. However, adding these axioms turned the ontology inconsistent again as some individuals were inferred to instantiate both PERSON and PUBLICATION. The reason for this inconsistency was an incorrect use of the editor relationship in SWRC. Although its domain was restricted to PERSON (editor_of), the property was apparently conceived to have *has_editor* semantics by most of the annotators. We fixed this inconsistency by changing the definition of editor accordingly. Another problem became apparent after we had already started the exploration of the resulting ontology with RELExO. An individual (in our opinion) belonging to the class RESEARCHPAPER was proposed as a counterexample, but could not be classified as such. A closer look at both individual and ontology showed that it was assigned to the class INPROCEEDINGS which was declared disjoint from RESEARCHPAPER, the latter actually being empty. Since we found that this modeling decision is not justified by the associated comments in the ontology, we simply removed the disjointness axiom.

Refinement Process.   In order to demonstrate the use of RELExO, we assume that we would like to add a new class REVIEWER to the SWRC ontology. Part of a change request could be a natural language description of this class such as *"a reviewer is a person who reviews a paper that has been submitted to a conference or workshop"*. Given this definitory sentence, LExO (cf. Section 4.3) automatically suggests an axiomatization of REVIEWER to the user who can correct or remove some of the generated axioms before they are added to the ontology.

REVIEWER $\equiv$ PERSON $\sqcap$ $\exists$review. $\big($PAPER $\sqcap$ $\exists$submitted_to. $($CONFERENCE $\sqcup$ WORKSHOP$)\big)$

Applying FOAM for suggesting mappings between the newly introduced class names and those already present in SWRC, we find PAPER to be equivalent to RESEARCHPAPER and add a corresponding equivalence axiom to the extended ontology. Likewise, we find PERSON, CONFERENCE and WORKSHOP already present in the original ontology.

In the next step, the set of "relevant" classes has to be selected. As mentioned in Section 4.4.2, it is reasonable to choose those atomic classes present in the definition of REVIEWER. We decided to add two more classes denoting undergraduate and PhD students and (introducing abbreviations for overly long concept names from $\mathcal{KB}$') we set:

$$\mathbf{C}' := \{\bot, \text{CoW}, \text{CONFERENCE}, \text{SubCoW}, \text{PERSON}, \text{PHDSTUDENT},$$
$$\text{RESEARCHPAPER}, \text{REVPSUBCOW}, \text{UNDERGRADUATE}, \text{WORKSHOP}\}.$$

Based on this set of classes, the RE algorithm is started. The first hypothetical DL axiom, the exploration comes up with is $\top \sqsubseteq \bot$. Naturally, this hypothesis cannot be deduced from the ontology. Hence, following the description in Section 4.4.2, KAON2 will query the knowledge base for instances of $\top \sqcap \neg\bot$ which is equivalent to $\top$. Hence *all* ABox individuals are retrieved. Choosing one of the retrieved individuals, in our case *id1289instance*, we find it to be an instance of RESEARCHPAPER and (since in our example,we chose the option to give the expert the opportunity to enhance the counterexample specification) add the information that it is an instance of SUBCOW.

In a similar way, the next hypothesis posed – $\top \sqsubseteq$ RESEARCHPAPER $\sqcap$ SUBCOW – is handled. Clearly, not every ABox individual is a research paper witnessed by the counterexample *id1303instance* being a journal article and hence neither a research paper (according to the underlying ontology) nor submitted to a conference or workshop.

However, the subsequent hypothesis CoW $\sqsubseteq \bot$ can neither be proved nor disproved by KAON2 using the information actually present in the ontology – since it does not contain any individuals being a conference or workshop. Therefore, the human expert will be asked for the final decision. Obviously, this hypothesis has to be denied and a counterexample for it is just any conference, so we enter *ICFCA_2008* and specify it as instance of CONFERENCE.[16] Note that due to the capability of dealing with partial information, the expert

---

[16]This information already qualifies *ICFCA_2008* as a counterexample for the presented hypothesis. RELExO checks for every
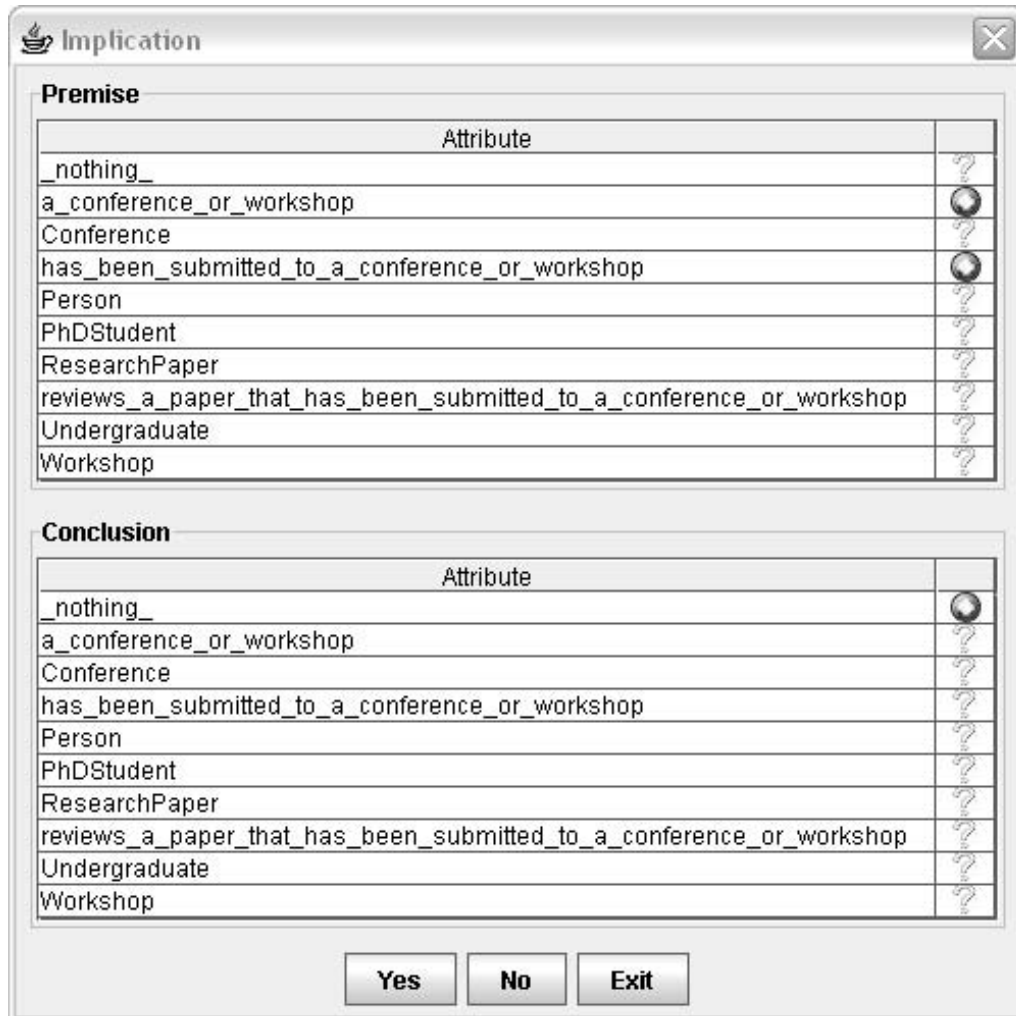
Figure 4.9: Dialog for displaying the hypothetical axiom CoW $\sqcap$ SubCoW $\sqsubseteq \perp$.

may leave open whether this individual belongs to the other considered classes. However, we employ the reasoner in order to determine all class-memberships deducible from the present information. In our case, it can be inferred that *ICFCA_2008* is also an instance of COW and definitely no instance of RESEARCHPAPER.

Consequently, the next question COW ⊑ CONFERENCE comes up and has to be denied as well by entering the workshop instance *OntoLex_2007*.
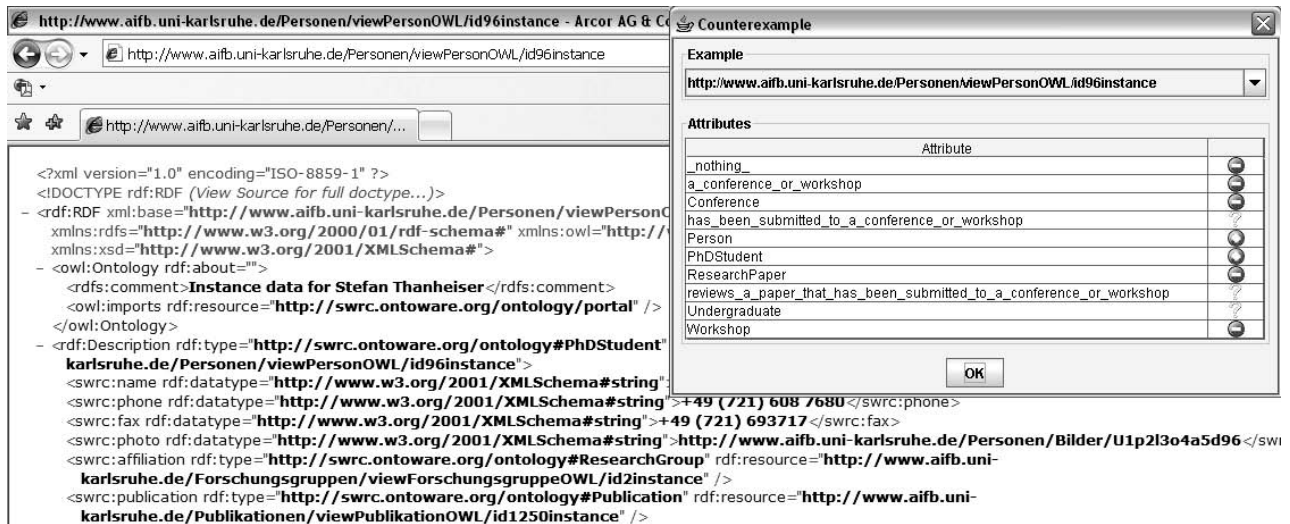
Figure 4.10: Specifying a counterexample. Every (non-)class-membership deducible from the knowledge base is automatically entered leaving just the open questions to the expert. RELExO can be configured to automatically display the web page associated with an individual's URI.

Equally, the hypothesis SUBCOW ⊑ RESEARCHPAPER cannot be decided based on the present knowledge and is thus passed to the expert. In fact, this is the first "design decision" to make depending on the intended scope of the ontology. A look into the SWRC taxonomy reveals that there is a class POSTER to denote posters presented at conferences. Indeed, any submitted poster would be a counterexample for the presented hypothesis, so we add *iMapping_Poster_SWUI_2006* to the knowledge base.

The next hypothesis brought up is COW ⊓ SUBCOW ⊑ ⊥ being an integrity constraint saying that nothing being a conference or workshop can be submitted (to a conference or workshop). Figure 4.9 shows how it is presented to the user. Here, we encounter another design decision. Although it might be reasonable to say that a workshop (actually: a workshop proposal) has been submitted to a conference, we stick to the intended semantics of the term *Workshop* as a kind of event which cannot be submitted and hence confirm the validity of the presented hypothesis.

The hypothesis PERSON ⊑ ⊥, coming up next, is refuted by the reasoner retrieving an individual who is a PhD student at the institute AIFB. Figure 4.10 shows the dialog wherein the user is presented the stored information about this individual and is asked to add the missing facts.

In this way, the exploration continues. During the process, some individuals are added and the following new axioms are confirmed:

- SUBCOW ⊓ PERSON ⊑ ⊥ (a person cannot be submitted)

- RESEARCHPAPER ⊑ SUBCOW (every research paper has been submitted to a conference or workshop)[17]

---

alleged counterexample whether it is indeed a such and rejects the input otherwise.

[17]We regard this justified by the existence of a class UNPUBLISHED disjoint to RESEARCHPAPER.

- RevPSubCoW $\sqsubseteq$ Person (everybody reviewing a submitted paper is a person)

- Person $\sqcap$ PhDStudent $\sqcap$ Undergraduate $\sqsubseteq$ $\bot$ (PhD students are disjoint with undergraduates)[18]

- RevPSubCoW $\sqcap$ Undergraduate $\sqcap$ Person $\sqsubseteq$ $\bot$ (actually a "policy decision": undergraduates are not allowed to review papers)

Figure 4.11: Partial formal context resulting from the exploration.

The formal context with the examples acquired during the exploration is displayed by Figure 4.11. It is automatically exported to the native ConExp[19] format and stored as a `CEX` file.

We end up with a refined SWRC ontology containing the new class Reviewer fully integrated into the existing ontology. Any subsumption between conjunctions of the specified interesting classes can be directly

---

[18]Another modeling flaw: this axiom should have been present in SWRC.

[19]http://conexp.sourceforge.net

decided based on this refined SWRC ontology. This can be nicely demonstrated by starting RELExO again with the refined ontology: it terminates without ever asking the human expert for a decision, showing that all upcoming questions can be answered by the reasoner alone.

4.6 Conclusion

In this chapter, we have shown the principal feasibility of an systematic lexico-logical enrichment of ontologies. By embedding our lexical approach the automatic acquisition of class descriptions (LExO) into a framework for reasoner-aided relational exploration (RELExO), we provide ontology engineers with an intuitive, yet efficient tool for expressive knowledge acquisition.

Chapter 5

Modelling Patterns for Ontology Engineering

As described in NeON Deliverable D5.1.1 (Modelling Components) and D2.5.1 (A Library of Ontology Design Patterns) there exist methods for introducing reuse in ontology design and additionally ontology modelling patterns can be utilised as a tool to support modelling teams. In this deliverable the focus is on how patterns are a part of the ontology context (see Section 1.1 for a definition of the notion of context), of how to interpret the ontology. In this sense patterns can be seen as background knowledge about either the particular logical language used (in the case of structural patterns) or common-sense knowledge about how to conceptualise certain situations (in the case of content patterns). As such, patterns are an important aid throughout the complete ontology life-cycle. Patterns can aid in constructing, evaluating, documenting and maintaining ontologies as well as improving the user understanding of the modelling problems and language specifics and aid communication and education of domain experts or novice ontology engineers. Although the actual benefits in terms of patterns as knowledge reuse are still to be measured and studied in detail (considered in NeOn WP5 experiments on ontology patterns).

Most modelling patterns are so far used manually, intended as an inspiration and source of encoded modelling experience rather than a detailed template solution, but patterns can also be used semi-automatically. When considering the current state-of-the-art in Ontology Learning (OL), as we have seen in the previous chapters this is much focused on discovering specific primitives from a certain input (for example natural language texts). If modelling patterns can be used semi-automatically they can improve the previously described methods for OL and combine them with encoded best-practises and template solutions to build better and more reasonable ontologies. For example ontology content patterns might help integrate diverse extraction results from different OL methods, refine an existing model already present or evaluate an existing or learnt ontology with respect to the encoded best-practises. The OntoCase approach described in Section 5.4 attempts to utilise ontology patterns semi-automatically to improve the result of other OL approaches. OntoCase is a pattern-based approach, related to the ontology design pattern activities ongoing in WP5 but is not presented as a result in that work package, instead it is introduced here due to its closer relation to OL methods and its current focus on matching between patterns and learnt ontologies (putting learnt ontologies into a context).

In Section 5.1 ontology patterns are related to the notion of context and in Section 5.2 the background, origin and types of ontology engineering patterns are described in brief, more information can be found in NeOn D5.1.1 and D2.5.1 (the sections here are only included to give the reader the background necessary to understand the subsequent sections). Next, in Section 5.3 the task of aligning a learnt ontology to top-level ontologies is described and discussed, and in Section 5.4 the OntoCase approach semi-automatically exploiting patterns in ontology construction is introduced. In Section 5.5 the chapter is summed up and some conclusions are drawn.

5.1   Context Sensitivity for Networked Ontologies

The notion of ontology patterns and the usage of such patterns, for ontology design or for example alignment of ontologies to top-level ontologies, is related to the context-sensitivity of ontologies in several ways.

Ontology engineering patterns.   When ontology engineering patterns are used on top of other OL techniques (as is suggested in the OntoCase approach described further in Section 5.4) the patterns represent a set of best-practises of the community. Thereby relating a learnt ontology to a pattern and applying the pattern is in some sense to relate the learnt ontology to the modelling best-practises. Content patterns additionally to some extent encode domain independent common-sense knowledge that can be included in the constructed ontology by applying the pattern.

Alignment to top-level ontologies.   In many cases the ontology patterns are "pieces" extracted from a larger construct, like a top-level ontology. In this case, relating to the pattern additionally means relating to the top-level ontology from which the pattern was extracted. A top-level ontology provides the basic definitions and axiomatisation that puts the learnt ontology into a context. Different top-level ontologies might result in different interpretations of the learnt ontology. Whether or not the alignment is always desirable as a consequence of applying a pattern can be discussed, since patterns may originate in different top-level ontologies and this may yield contradictory interpretations. Such problems are part of the evaluation and revision phase in the OntoCase approach suggested in Section 5.4, but this is still future work.

5.2   Ontology Engineering Patterns

Most researchers agree that patterns can act as good guidelines for inexperienced designers and additionally assist the communication between both designers and from designers to users or software maintainers. The main dispute is whether there is really an additional reuse benefit of patterns in general, if a system really becomes "better" in some sense by reusing a proven solution (see for example [BCC+96], [Men97], [DFAM02] and [PUPT01]). Still, patterns are widely used in practise and analogous to the pattern community in software engineering also the knowledge engineering community has adopted the idea of patterns, inspired by for example the idea of scripts in Artificial Intelligence (see [RS89]), components for problem solving methods like in the KADS methodology (see [GRC+98]), data model patterns for database design as introduced in [Hay96], and the knowledge patterns presented in [CTP00]. A thorough state-of-the-art and more background can be found in NeOn deliverable D5.1.1, and an up to date catalogue of patterns in D2.5.1.

Ontology design patterns (OPs) are, according to D2.5.1, divided into the following categories:

- Structural OPs

- Correspondence OPs

- Content OPs

- Reasoning OPs

- Presentation OPs

- Lexico-syntactic OPs

Structural patterns can be logical or architectural OPs. Logical patterns propose solutions for design problems (expressivity problems) of a specific modelling language (e.g. OWL), independently of a particular conceptualisation, thus addressing logical problems. Architecture patterns are a specific kind of logical patterns

addressing the overall structure of the intended ontology, such a pattern may for example be a composition of a set of logical patterns. Content patterns propose solutions (in OWL or another logical language) to design problems for the domain classes and properties that constitute an ontology, thus addressing content problems. In this deliverable no in-depth description of logical, architecture and content patterns are presented, the reader is referred to deliverables D5.1.1 and D2.5.1 for further details, and the remaining types of patterns are not treated at all here.

5.2.1 Structural OPs

A logical pattern is a content-independent structure, untyped and expressed only with logical vocabulary. Such idioms, or logical ontology design patterns, have recently been the subject of a W3C task force, working within the Semantic Web Best Practices and Deployment Working Group. The group developed patterns for common constructs in OWL (see [WS04]). An illustration of such a pattern representing one way to express n-ary relations in OWL, found in [NR], can be seen in Figure 5.1. The illustration shows an implementation of this pattern, stating that an n-ary relation can be replaced by a new concept, using the example sentence "Anne has lung cancer with high probability".
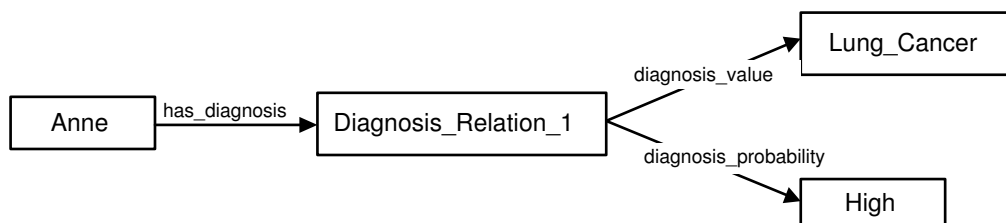
Figure 5.1: A logical pattern for representing N-ary relations in OWL. [NR]

In the NeOn project an initial list of 16 logical patterns for OWL have been proposed in Deliverable D5.1.1. These patterns include simple patterns like primitive class, defined class and subClassOf relation and more advanced logical patterns like exhaustive classes and the n-ary relations pattern illustrated above. Many of the logical patterns are already implemented and used implicitly in the modelling capabilities of the NeOn toolkit for ontology engineering.

Architecture patterns for ontologies are a kind of logical patterns that describe the overall structure of the ontology. Usually an architecture pattern describe some kind of restrictions on the way the ontology should be modelled, in order to conform to the pattern. An ontology might be constructed as a tree of categories, the tree structure could be an example of an architecture pattern for ontologies (probably a taxonomy). Other such patterns are related to the idea of modularisation, for example the kind of modules and structures described in [Rec03], where each module has its own specific structural properties.

In the NeOn project an initial list of 3 architectural design patterns for ontologies have been proposed in Deliverable D5.1.1. These patterns include the taxonomy pattern, light-weight ontologies and a modular structure.

5.2.2 Content OPs

A content pattern is an instance of a logical pattern, or a composition of several logical patterns. A content pattern is a typed structure, e.g. including non-logical (possibly domain-specific) vocabulary. A content design pattern represents and solves a specific modelling problem, but is restricted to solve a small subproblem, in contrast to the architecture patterns in the previous section that describe the logical structure of the complete ontology.

Recent research has been presented concerning content patterns, for example the ontology content patterns described in [Gan05]. Many of these patterns have been extracted from top-level ontologies, like DOLCE (see

description in D5.1.1 and D2.5.1). An example of such a pattern is the participation pattern, describing how objects may participate in events as illustrated in Figure 5.2.
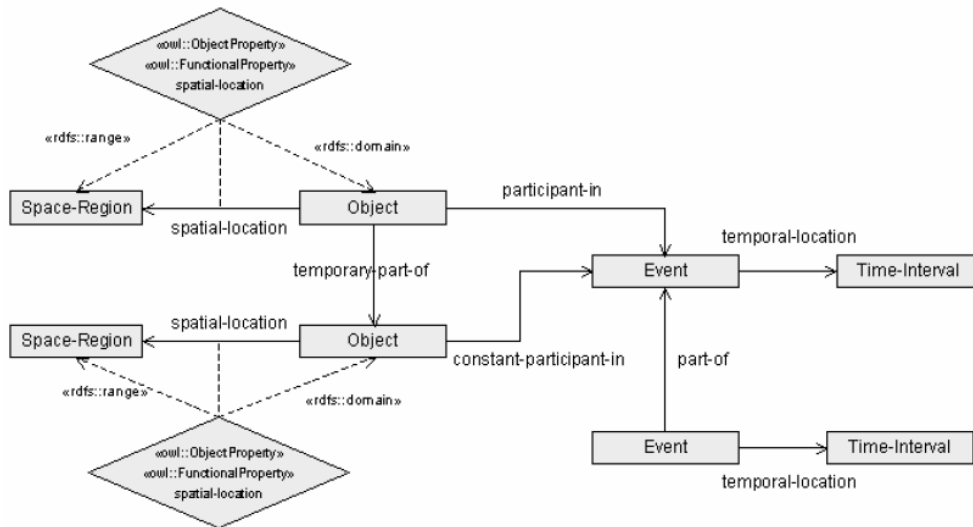

Figure 5.2: A content pattern describing participation of objects in events (see D5.1.1).

In the NeOn project an initial list of 5 content design patterns for OWL ontologies have been proposed in Deliverable D5.1.1, and a more extensive catalogue of 34 content OPs in D2.5.1. The patterns in D5.1.1 include the participation pattern mentioned above, along with the descriptions-situations pattern, role-tasks pattern, plan-execution pattern and the simple part-whole relation patterns. These patterns have implicitly been used to evaluate and discuss the ontologies produced in the NeOn use-cases so far.

5.3   Alignment to Top-level Ontologies

Alignment of learnt or manually engineered ontologies to top-level ontologies is a special case of ontology matching (as for example defined in [ES07]), where the intention is to primarily find correspondences between more general concepts or relations in the top-level ontology and more specific concepts and relations in the engineered ontology.

In theory the same methods as for generic ontology matching could be used (see survey in [ES07]), but some will be more or less useful for this special case. For example, looking for direct overlap between the ontologies is not very fruitful, since the learnt ontology and the top-level ontology are usually on very different levels of abstraction. In the case of a learnt ontology additionally the connection back to the information used as input for the construction (usually a text corpus) can be utilised for the matching. A learnt ontology though will probably be harder to align due to its diverse nature and uncertainty. Additionally, methods commonly applied for OL (like learning disjointness axioms as described in Chapter 3) can be used to improve the quality of such an alignment.

To find subsumption relations between concepts in the current ontology and some top-level ontology, several methods have been proposed in literature. A widely used approach is to exploit some existing background knowledge, like domain taxonomies or language resources such as WordNet to try to find subsumption relations between concepts in the two ontologies. Additionally similar techniques as for ontology learning can be used to try and connect concepts (see chapter 2), but this time not only using the text corpus at hand but exploiting external resources like the web to find valid relations (as in [IBCW06] for example). Of course this might give rise to other problems, like disambiguation, which then have to be solved by determining a confidence value for the correspondence.

Aligning an ontology to a top-level ontology might also be compared to automatically specialising or extending a top-level ontology. In this case, methods like lexical substitution might be used to find clues of whether or not a more general concept is related to a more specific one in the other ontology. Lexical substitution techniques might also be used as a way to verify hypotheses to increase the accuracy (as in [GPP07]), rather than extracting relations, since trying all possible combinations might be too time consuming.

The alignment of an ontology to a top-level ontology might also be done through the matching of ontology engineering patterns, in this case primarily content patterns, which in many cases already have a connection (or are extracted from) a top-level ontology. By determining that a pattern can be applied and applying it then provides a connection to the top-level ontology, as will be described for the OntoCase approach below.

5.4   OntoCase

OntoCase is one proposed approach to use ontology patterns throughout an iterative ontology construction and evolution framework (presented in [Blo07]). The framework can be viewed as a general methodology applied on top of more traditional OL methods (see previous chapters) and it uses the OL from text as an initial step to gather input for the pattern matching, specialisation and adaptation. OntoCase is inspired by the case-based reasoning methodology (see [AP94]), where usually partial reusable solutions are stored and then retrieved, adapted and composed when a new solution is requested. In OntoCase patterns constitute the backbone of these reusable solutions, in the sense that patterns can be utilised directly as solutions to specific modelling problems but can also be connected to more specific reusable components through specialisations and variant relations.
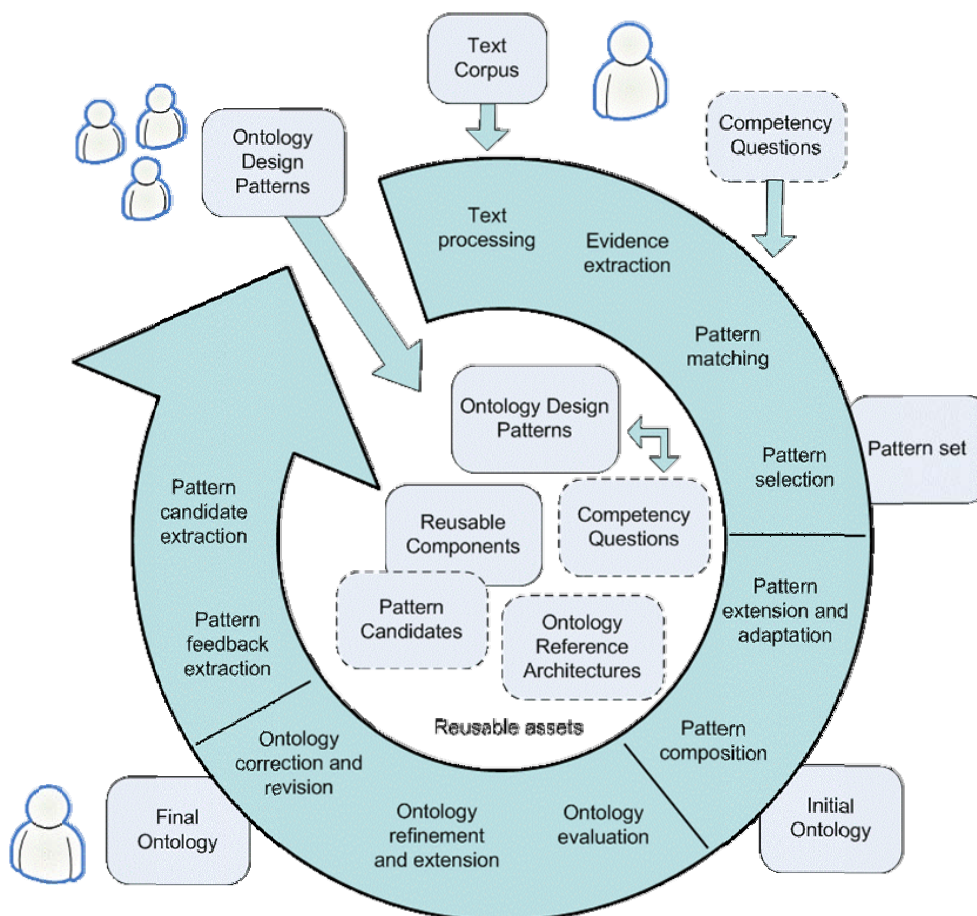

Figure 5.3: Overview of the OntoCase cycle.

In the OntoCase approach the central repository of reusable components consists of a pattern catalogue, containing both ontology content patterns, ontology architecture patterns (tentatively, although no such patterns have been used in practise yet) and other reusable assets (that may be specialisations of patterns, pattern candidates or other reusable components retrieved from constructed solutions). For examples of ontology content and architecture patterns, see previous sections or D5.1.1 and D2.5.1, and an attempt to describe the area of ontology patterns to which OntoCase is related can be found in [BS05]. Additionally it is envisioned that the patterns are enriched with corresponding competency questions, to in the future improve pattern retrieval and adhere more closely to the case-based reasoning paradigm, but so far all matching is based on the pattern structures themselves.

The OntoCase cycle consists of four phases, analogous to a traditional CBR methodology, that may be run in sequence but may also be considered as self contained processes that can be run and re-run as long as the required input is provided. The first OntoCase phase (retrieval) corresponds to input analysis and pattern retrieval (the first four steps in Figure 5.3). It constitutes the process of analysing the input (so far only text corpora have been used) and matching the derived input representation to the pattern base, to select appropriate patterns. OntoCase is not to be considered a completely new OL approach, but merely adding a pattern application cycle on top of other proposed methods and additionally giving an overall framework for pattern-based OL. Thereby other OL approaches that process a text corpus may be used for the input processing. The assumption made in OntoCase is that such an approach will at least produce a set of terms and their confidence levels, as well as a set of binary relations (named or unnamed) and their confidence levels. These are the two categories of input that the pattern-matching algorithms so far take into account, but this could of course be extended as future work.

The challenge is then to bridge the gap between the abstract patterns and the specific terms and relations extracted from text. The pattern selection process has been based on methods similar to existing approaches in ontology matching and ontology ranking, but tailored to fit the specific case of pattern ranking and selection. The ranking algorithm (described in depth in [Blo08]) used is based on four measures: concept coverage, relation coverage, density and proximity. Concept coverage is computed using three measures, including direct string matching and subsumption coverage (using WordNet and a "head" heuristic directly applied on the concept labels). Relation coverage uses the concept to term matches from the previous step, and tries to match both relation labels as well as domain and range of relations. Density and proximity are "weighting scores" that aim to discriminate patterns that seem less useful after the first matching of concepts and relations have been performed. The intuition is that a pattern will be more useful to specialise and adapt if a larger and more coherent part of the pattern is detected in the input.

The second phase (reuse) includes pattern specialisation, adaptation and composition, and constitutes the process of reusing retrieved patterns and constructing an improved ontology (the two following steps in Figure 5.3). The basis is the input representation, the ontology extracted by "conventional" OL methods as described above. Patterns can then be specialised using the matching information produced in the first phase, adapted through for example pruning and specialisation, and finally composed into an ontology. Pattern specialisation is achieved primarily on the basis of the matching information of the extracted terms and their relations and composition utilises additionally the connections between patterns (for example through a top-level ontology). Whether all information in the extracted input representation is used or not may be a trade-off between coverage of the information of the input text corpus and connectedness and coherence of the constructed ontology, depending on the quality of the initially extracted ontology.

The third phase concerns evaluation and revision of the ontology to improve the fit to the input and the ontology quality. Although logical patterns may have been used implicitly also in previous phases, in this phase both logical and content patterns may be used for evaluating the ontology. Major issues are to improve consistency and reduce redundancy that may have been introduced during pattern composition. The final phase includes the discovery of new pattern candidates or other reusable components as well as storing pattern feedback. Pattern extraction is related to issues like module detection and strongly connected components in graphs, but such approaches for pattern detection is still future work.

So far the OntoCase approach is only partially implemented. The pattern catalogue is realised using a database of patterns and other reusable components, and the two first phases of the method are then implemented on top of the Jena Semantic Web framework[1] and incorporating interfaces to tools like SimMetrics[2] for string matching and WordNet (as described in [C. 98]) for relation and synonym identification. So far the application has no graphical user interface, but this is of course a future work step. Additionally, only partial evaluations have been performed (see previously referenced papers) and an earlier version of the approach was used in a research project in cooperation with industry, as presented in [BOS06].

5.5   Conclusion

Ontology patterns is a popular research area, since patterns have proven useful in many other areas the same ideas are also entering the ontology engineering community. Still, it is not obvious that patterns really give benefits at all times, and this will have to be proven scientifically in the future in order to develop more detailed guidelines when to use patterns and better tuned methods for applying them. At the moment patterns are used mainly manually, but as we have described in this chapter also semi-automatic application of patterns is possible. To apply a pattern helps to put the learnt ontology into a context, and possibly additionally align it to a top-level ontology.

The OntoCase approach introduces a pattern application step on top of other OL methods, as described in previous chapters. The intention is to produce better quality of the ontologies and to reduce the load on the human user. Still, there are many improvements possible in this approach. Major improvements would be to also incorporate competency questions in the pattern matching process, to take into account the intention of the ontology and the patterns and not only the primitives of the patterns and the primitives extracted by OL methods from text. Additionally general architecture patterns should guide the pattern composition step and restrict the composition of content patterns. Also interesting research opportunities lie in the last steps of the process, automatically extracting new patterns, which can then be used when constructing new ontologies.

---

[1] http://jena.sourceforge.net
[2] http://www.dcs.shef.ac.uk/~sam/simmetrics.html

Chapter 6

Conclusion and Outlook

In this deliverable, we presented a variety of different methods and tools, that we envision as core components of a framework for semi-automatic ontology engineering, i.e. ontology engineering supported by automatic means for ontology acquisition, evaluation and refinement. Many of our tools such as Text2Onto and LeDA have already been implemented as plugins for the NeOn Toolkit (see Sections 2.4 and 3.5), even more are supposed to follow within the next weeks and months.

- **Text2Onto** (Chapter 2 is a framework for basic ontology learning, that has been developed to support the acquisition of lightweight ontologies.

- **LeDA** (Chapter 3) automatically enriches a given ontology with disjointness axioms, thus complementing the basic ontology learning methods of Text2Onto. In a way, LeDA also facilitates the evaluation of lightweight taxonomies, since logical inconsistencies caused by learned disjointness axioms can help to reveal particular modeling errors.

- **RELExO** (Chapter 4) enables a more detailed refinement and evaluation of ontologies. While its LExO component assists the ontology engineer in the process of axiomatizing atomic classes, e.g. introduced by Text2Onto, the exploration part helps to integrate newly acquired entities into the ontology. It also helps the user to detect "inconsistencies" or mismatches between the ontology and her conceptualization, and facilitates an intuitive, stepwise approximation of the user's domain knowledge.

- **OntoCase** (Chapter 5) is a framework supporting the refinement of learned or manually engineered ontologies by means of ontology patterns. It also enables an alignment with top-level ontologies, hence facilitating both ontology evaluation and mapping.

Additional tools, not described in detail here, but potentially relevant for a framework as envisioned include:

- **AEON** [VVS05] is a framework for the automatic evaluation of ontologies with respect to the OntoClean methodology. It automatically suggests a set of meta-property taggings and checks the taxonomic hierarchy for OntoClean constraint violations.

- **RoLExO** (*submitted*) can be seen as an extension of RELExO (see Chapter 4), which specifically supports the acquisition of complex property restrictions.

- **RaDON**[1] is a tool for inconsistency diagnosis and repair. Its debugging functionalities can be required in case logical inconsistencies are introduced by manual or automatic ontology acquisition methods.

- **FOAM** [ES05], a framework for ontology alignment, could facilitate the integration of different ontology learning results into an existing ontology.

---

[1]`http://radon.ontoware.org`

By integrating these tools into a common platform we hope to facilitate the interplay of components for both manual and automatic ontology engineering, hence to increase the efficiency of the overall knowledge acquisition process. We consciously chose the term "semi-automatic ontology engineering" (as opposed to ontology learning, for example), since it puts a stronger emphasis on methodological and interactive aspects, which we consider essential for the success of future knowledge acquisition approaches.

In the future, we plan to identify (and possibly implement) further components to complement the proposed framework for semi-automatic ontology engineering. We will analyse different use cases and application requirements, in order to develop an integrated evaluation scenario.

Bibliography

[AG06]   M. Amoia and C. Gardent. Adjective based inference. In *Proceedings of the EACL Workshop on Knowledge and Reasoning for Answering Questions (KRAQ'06)*, April 2006.

[AP94]   A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues , methodological variations, and system approaches. *AICom*, 7:39–59, 1994. IOS Press.

[BADW04]   C. Brewster, H. Alani, S. Dasmahapatra, and Y. Wilks. Data Driven Ontology Evaluation. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Lisbon, Portugal, 2004.

[BCC$^+$96]   Kent Beck, James O. Coplien, Ron Crocker, Lutz Dominick, Gerard Meszaros, Frances Paulisch, and John Vlissides. Industrial experience with design patterns. In *Proceedings of the 18th International Conference on Software Engineering*. IEEE Computer Society Press, 1996.

[Blo07]   Eva Blomqvist. Ontocase - a pattern-based ontology construction approach. In *Proccedings of OTM 2007: ODBASE - The 6th International Conference on Ontologies, DataBases, and Applications of Semantics*, Vilamoura, Algarve, Portugal, November 25-30 2007.

[Blo08]   Eva Blomqvist. Pattern ranking for semi-automatic ontology construction. In *In: Proceedings of SAC 2008 - Semantic Web Applications track*, Fortaleza, Brazil, March 2008.

[BOS06]   E. Blomqvist, A. Öhgren, and K. Sandkuhl. Ontology Construction in an Enterprise Context: Comparing and Evaluating two Approaches. In *Proc. of ICEIS'06*, Paphos, Cyprus, May 2006.

[BS05]   E. Blomqvist and K. Sandkuhl. Patterns in Ontology Engineering: Classification of Ontology Patterns. In *Proc. of ICEIS2005*, Miami Beach, Florida, May 24-28 2005.

[Bur91]   Peter Burmeister. Merkmalimplikationen bei unvollständigem Wissen. In Wilfried Lex, editor, *Arbeitstagung Begriffsanalyse und Künstliche Intelligenz*, pages 15–46. TU Clausthal, 1991.

[C. 98]   C. Fellbaum et al. *WordNet - An Electronic Lexical Database*. MIT Press, 1998.

[CBH85]   M. Chodorow, R.J. Byrd, and G.E. Heidorn. Extracting semantic hierarchies from a large on-line dictionary. In *Proceedings of the 23rd annual meeting on Association for Computational Linguistics*, pages 299–304, 1985.

[CPSTS05] P. Cimiano, A. Pivk, L. Schmidt-Thieme, and S. Staab. Learning taxonomic relations from heterogeneous sources of evidence. In *Ontology Learning from Text: Methods, Applications and Evaluation*. IOS Press, 2005.

[CTP00]   P. Clark, J. Thompson, and B. Porter. Knowledge Patterns. In A. G. Cohn, F. Giunchiglia, and B. Selman, editors, *KR2000: Principles of Knowledge Representation and Reasoning*, San Francisco, 2000. Morgan Kaufman.

NeOn

[CV05a]   Philipp Cimiano and Johanna Völker. Text2onto - a framework for ontology learning and data-driven change discovery. In Andres Montoyo, Rafael Munoz, and Elisabeth Metais, editors, *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB)*, volume 3513 of *Lecture Notes in Computer Science*, pages 227–238, Alicante, Spain, JUN 2005. Springer.

[CV05b]   Philipp Cimiano and Johanna Völker. Towards large-scale, open-domain and ontology-based named entity classification. In G. Angelova, K. Bontcheva, R. Mitkov, and N. Nicolov, editors, *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP)*, pages 166–172, Borovets, Bulgaria, SEP 2005. INCOMA Ltd.

[DFAM02]   Andy Dearden, Janet Finlay, Liz Allgar, and Barbara McManus. Evaluating pattern languages in participatory design. In *Proceedings of CHI2002*, Minneapolis, USA, April 2002. ACM.

[ES05]   Marc Ehrig and York Sure. FOAM - framework for ontology alignment and mapping. results of the ontology alignment initiative. In Benjamin Ashpole, Marc Ehrig, JÃŠrÃ´me Euzenat, and Heiner Stuckenschmidt, editors, *Proc. of the Workshop on Integrating Ontologies*, volume 156, pages 72–76, OCT 2005.

[ES07]   Jerome Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer Berlin Heidelberg, 2007.

[FBS07]   Baris Sertkaya Franz Baader, Bernhard Ganter and Ulrike Sattler. Completing description logic knowledge bases using formal concept analysis. In Manuela M. Veloso, editor, *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 230–235, JAN 2007.

[Fel98]   Christiane D. Fellbaum. *WordNet – An Electronic Lexical Database*. MIT Press, 1998.

[Gan99]   Bernhard Ganter. Attribute exploration with background knowledge. *Theoretical Computer Science*, 217(2):215–233, 1999.

[Gan05]   A. Gangemi. Ontology Design Patterns for Semantic Web Content. In *Proceedings of ISWC 2005*, volume 3729 of *LNCS*, pages 262–276. Springer, 2005.

[GHVD03]   Benjamin Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description logic programs: Combining logic programs with description logics. In *Proc. of the International World Wide Web Conference (WWW'03)*, pages 48–57. ACM, MAY 2003.

[GPP07]   Alfio Massimiliano Gliozzo, Marco Pennacchiotti, and Patrick Pantel. The domain restriction hypothesis: Relating term similarity and semantic consistency. In *Proceedings of North American Association for Computational Linguistics / Human Language Technology (NAACL HLT 07)*, Rochester, NY, 2007.

[Gra90]   R. M. Gray. *Entropy and Information Theory*. Springer, 1990.

[GRC$^+$98]   K. Gardner, A. Rush, M. Crist, R Konitzer, and B. Teegarden. *Cognitive Patterns - Problem-solving Frameworks for Object Technology*. Cambridge University Press, 1998.

[GSWB90]   L. Guthrie, B.M. Slator, Y. Wilks, and R. Bruce. Is there content in empty heads? In *Proceedings of the 13th conference on Computational linguistics*, pages 138–143, Morristown, NJ, USA, 1990. Association for Computational Linguistics.

[Har54]   Zellig Sabbetai Harris. Word. *Distributional Structure*, 10(23):146–162, 1954.

[Hay96]   D. C. Hay. *Data Model Patterns - Conventions of Thought*. Dorset House Publishing, 1996.

[Hea92]   M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics*, pages 539–545, 1992.

[IBCW06]   Jose Iria, Christopher Brewster, Fabio Ciravegna, and Yorick Wilks. An Incremental Tri-partite Approach to Ontology Learning. In *Proc. of LREC2006*, Genoa, May 2006.

[Jac12]   Paul Jaccard. The distribution of flora in the alpine zone. 11:37–50, 1912.

[KF98]   J. Tsuji K. Frantzi, S. Ananiadou. The c-value/nc-value method of automatic recognition for multi-word terms. In *Proceedings of the ECDL*, pages 585–604, 1998.

[KPP03]   J. Klavans, S. Popper, and B. Passonneau. Tackling the internet glossary glut: Automatic extraction and evaluation of genus phrases. In *Proceedings of the SIGIR'03 Workshop on Semantic Web*, 2003.

[Lin98]   Dekang Lin. Dependency-based evaluation of MINIPAR. In *Proceedings of the Workshop on the Evaluation of Parsing Systems*, May 1998.

[Men97]   Tim Menzies. Object-oriented patterns: Lessons from expert systems. *Software - Practice and Experience*, 1(1), December 1997.

[MS00]   A. Maedche and S. Staab. Discovering conceptual relations from text. In W. Horn, editor, *Proceedings of the 14th ECAI'2000*, 2000.

[MV01]   Alexander Mädche and Raphael Volz. The Text-To-Onto ontology extraction and maintenance system. In *Workshop on Integrating Data Mining and Knowledge Management, collocated with the 1st International Conference on Data Mining (ICDM)*, 2001.

[MVS08]   Christian Meilicke, Johanna Völker, and Heiner Stuckenschmidt. Learning disjointness for debugging mappings between lightweight ontologies. 2008. submitted to ESWC.

[NR]   N. Noy and A. Rector. Defining N-ary Relations on the Semantic Web: Use With Individuals. W3C Working Draft 10 June 2004, available at: http://www.w3.org/2001/sw/BestPractices/.

[PP03]   Banerjee Patwardhan and Pedersen. Using measures of semantic relatedness for word sense disambiguation. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*, pages 241–257, FEB 2003.

[PUPT01]   Lutz Prechelt, Barbara Unger, Michael Philippsen, and Walter Tichy. Two controlled experiments assessing the usefulness of design patterns documentation in program maintenance. *IEEE TRansactions on Software Engineering*, 2001.

[Qui93]   J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[Rec03]   A. Rector. Modularisation of Domain Ontologies Implemented in Description Logics and related formalisms including OWL. In *Proceedings of the international conference on Knowledge capture*, pages 121–128, Sanibel island, 2003. ACM Press.

[RS89]   C. Riesbeck and R. Schank. *Inside Case-based Reasoning*. Lawrence Erlbaum Associates Inc., 1989.

[Rud04]   Sebastian Rudolph. Exploring relational structures via FLE. In Karl Erich Wolff, Heather D. Pfeiffer, and Harry S. Delugach, editors, *Conceptual Structures at Work: 12th International Conference on Conceptual Structures*, volume 3127 of *LNCS*, pages 196–212, Huntsville, AL, USA, JUL 2004. Springer.

[Rud06]   Sebastian Rudolph. *Relational Exploration - Combining Description Logics and Formal Concept Analysis for Knowledge Specification*. Universitätsverlag Karlsruhe, DEC 2006. Dissertation.

[Sal91]   G. Salton. Developments in automatic text retrieval. 253:974–979, 1991.

[SBH⁺05]   York Sure, Stephan Bloehdorn, Peter Haase, Jens Hartmann, and Daniel Oberle. The SWRC ontology - semantic web for research communities. In Carlos Bento, Amilcar Cardoso, and Gael Dias, editors, *Proc. of the 12th Portuguese Conference on Artificial Intelligence*, pages 218 – 231. Springer, DEC 2005.

[SC99]   Mark Sanderson and W. Bruce Croft. Deriving concept hierarchies from text. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 206–213. ACM, August 1999.

[SVB⁺05]   Ondrej Svab, Svatek Vojtech, Petr Berka, Dusan Rak, and Petr Tomasek. Ontofarm: Towards an experimental collection of parallel ontologies. In *Poster Proceedings of the International Semantic Web Conference 2005*, 2005.

[VHC07]   Johanna Völker, Pascal Hitzler, and Philipp Cimiano. Acquisition of owl dl axioms from lexical resources. In Enrico Franconi, Michael Kifer, and Wolfgang May, editors, *Proceedings of the 4th European Semantic Web Conference (ESWC'07)*, volume 4519 of *Lecture Notes in Computer Science*, pages 670–685. Springer, JUN 2007.

[VNCN05]   P. Velardi, R. Navigli, A. Cuchiarelli, and F. Neri. Evaluation of OntoLearn, a methodology for automatic population of domain ontologies. In P. Buitelaar, P. Cimiano, and B. Magnini, editors, *Ontology Learning from Text: Methods, Applications and Evaluation*, number 123 in Frontiers in Artificial Intelligence and Applications, pages 92–106. IOS Press, 2005.

[VS05]   Johanna Völker and York Sure. Data-driven change discovery. Technical report, Institute AIFB, University of Karlsruhe, JUL 2005. SEKT Deliverable 3.3.1.

[VVS05]   Johanna Völker, Denny Vrandecic, and York Sure. Automatic evaluation of ontologies (AEON). In Y. Gil, E. Motta, V. R. Benjamins, and M. A. Musen, editors, *Proceedings of the 4th International Semantic Web Conference (ISWC2005)*, volume 3729 of *LNCS*, pages 716–731. Springer Verlag Berlin-Heidelberg, NOV 2005.

[VVS07]   Johanna Völker, Denny Vrandecic, and York Sure. Data-driven change discovery. Technical report, Institute AIFB, University of Karlsruhe, JAN 2007. SEKT Deliverable 3.3.3.

[VVSH07]   Johanna Völker, Denny Vrandecic, York Sure, and Andreas Hotho. Learning disjointness. In Enrico Franconi, Michael Kifer, and Wolfgang May, editors, *Proceedings of the 4th European Semantic Web Conference (ESWC'07)*, volume volume 4519 of *Lecture Notes in Computer Science*, pages 175–189. Springer, JUN 2007.

[WS04]   W3C-SWBPD. Semantic Web Best Practices and Deployment Working Group. Available at: http://www.w3.org/2001/sw/BestPractices/, 2004.