NeOn: Lifecycle Support for Networked Ontologies

Integrated Project (IST-2005-027595)

Priority: IST-2004-2.4.7 — “Semantic-based knowledge and content systems”

D1.5.4 Bottom-up evolution of networked ontologies from metadata

Deliverable Co-ordinator: **Diana Maynard**

Deliverable Co-ordinating Institution: **University of Sheffield (USFD)**

Other Authors: **Diana Maynard (USFD); Niraj Aswani (USFD); Simon Schenk (UKO-LD); Johanna Völker (UKARL); Fouad Zablith (OU)**

This deliverable describes the continued implementation of a variety of approaches for bottom-up ontology evolution from different sources: unstructured text, ontologies, folksonomies, databases and queries. These approaches are all implemented as plugins for the NeOn toolkit: either loosely or tightly coupled. Management of the ontology lifecycle is also aided by the inclusion of methods for ontology change management which enable the synchronisation of changes in the different tools with the NeOn toolkit. We describe also a tool for ontology refinement which can make use of the information from these different tools as further input to improving the ontology. The ontological changes generated are evaluated not only according to correctness but also to relevance in the ontology, and various suggestions are also proposed for further improvement.

Document Identifier:	NEON/2010/D1.5.4/v1.0	Date due:	January 31, 2010
Class Deliverable:	NEON EU-IST-2005-027595	Submission date:	January 31, 2010
Project start date	March 1, 2006	Version:	v1.0
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 Knowledge Media Institute – KMi Berrill Building, Walton Hall Milton Keynes, MK7 6AA United Kingdom Contact person: Martin Dzbor, Enrico Motta E-mail address: {m.dzbor, e.motta}@open.ac.uk	**Universität Karlsruhe – TH (UKARL)** Institut für Angewandte Informatik und Formale Beschreibungsverfahren – AIFB Englerstrasse 11 D-76128 Karlsruhe, Germany Contact person: Peter Haase E-mail address: pha@aifb.uni-karlsruhe.de
Universidad Politécnica de Madrid (UPM) Campus de Montegancedo 28660 Boadilla del Monte Spain Contact person: Asunción Gómez Pérez E-mail address: asun@fi.ump.es	**Software AG (SAG)** Uhlandstrasse 12 64297 Darmstadt Germany Contact person: Walter Waterfeld E-mail address: walter.waterfeld@softwareag.com
Intelligent Software Components S.A. (ISOCO) Calle de Pedro de Valdivia 10 28006 Madrid Spain Contact person: Jesús Contreras E-mail address: jcontreras@isoco.com	**Institut 'Jožef Stefan' (JSI)** Jamova 39 SL–1000 Ljubljana Slovenia Contact person: Marko Grobelnik E-mail address: marko.grobelnik@ijs.si
Institut National de Recherche en Informatique et en Automatique (INRIA) ZIRST – 665 avenue de l'Europe Montbonnot Saint Martin 38334 Saint-Ismier, France Contact person: Jérôme Euzenat E-mail address: jerome.euzenat@inrialpes.fr	**University of Sheffield (USFD)** Dept. of Computer Science Regent Court 211 Portobello street S14DP Sheffield, United Kingdom Contact person: Hamish Cunningham E-mail address: hamish@dcs.shef.ac.uk
Universität Koblenz-Landau (UKO-LD) Universitätsstrasse 1 56070 Koblenz Germany Contact person: Steffen Staab E-mail address: staab@uni-koblenz.de	**Consiglio Nazionale delle Ricerche (CNR)** Institute of cognitive sciences and technologies Via S. Marino della Battaglia 44 – 00185 Roma-Lazio Italy Contact person: Aldo Gangemi E-mail address: aldo.gangemi@istc.cnr.it
Ontoprise GmbH. (ONTO) Amalienbadstr. 36 (Raumfabrik 29) 76227 Karlsruhe Germany Contact person: Jürgen Angele E-mail address: angele@ontoprise.de	**Food and Agriculture Organization of the United Nations (FAO)** Viale delle Terme di Caracalla 00100 Rome Italy Contact person: Marta Iglesias E-mail address: marta.iglesias@fao.org
Atos Origin S.A. (ATOS) Calle de Albarracín, 25 28037 Madrid Spain Contact person: Tomás Pariente Lobo E-mail address: tomas.parientelobo@atosorigin.com	**Laboratorios KIN, S.A. (KIN)** C/Ciudad de Granada, 123 08018 Barcelona Spain Contact person: Antonio López 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:

- University of Sheffield
- Open University
- University of Karlsruhe
- University of Koblenz-Landau

Change Log

Version	Date	Amended by	Changes
0.1	10-10-2009	Johanna Völker	Initial setup
0.2	3-12-2009	Diana Maynard	First draft

Executive Summary

This deliverable describes the continued implementation of a variety of approaches for bottom-up ontology evolution from different sources: unstructured text, ontologies, folksonomies, databases and queries. Previously in WP1, we have described methods for top-down ontology evolution, which incorporate a method for modelling some of the dynamics of (semantic) metadata, and which focused particularly on the interaction between the example clients and the NeOn toolkit, such that changes in networked ontologies can be propagated to the semantic metadata, and vice versa. From the bottom-up point of view, we have described preliminary versions of several systems that enable new information to be captured in ontological form and propagated to the NeOn toolkit, using data from unstructured text and folksonomies, namely SPRAT, SARDINE, Evolva and FLOR. In WP3, a preliminary version of Evolva has also been described, which aims more at ontology refinement rather than direct ontology generation.

These approaches are all implemented as plugins for the NeOn toolkit: either loosely or tightly coupled. Management of the ontology lifecycle is also aided by the inclusion of methods for ontology change management which enable the synchronisation of changes in the different tools with the NeOn toolkit. We describe also a tool for ontology refinement which can make use of the information from these different tools as further input to improving the ontology. The ontological changes generated are evaluated not only according to correctness but also to relevance in the ontology, and various suggestions are also proposed for further improvement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Ontology generation and population is a crucial part of knowledge base construction and maintenance that enables us to relate text to ontologies, providing on the one hand a customised ontology related to the data and domain with which we are concerned, and on the other hand a richer ontology which can be used for a variety of semantic web-related tasks such as knowledge management, information retrieval, question answering, semantic desktop applications, and so on. Ontology evolution is also an important task for a number of reasons. First, the data on which ontologies are based may be constantly evolving. New relevant documents may become available based on additional knowledge acquired or new facts that emerge. Old data may become superseded because facts change: for example, the CEO of a company will not always be the same, a species that was once endangered may no longer be so, new production techniques may be developed, even the names of cities and countries may change over time. These facts have to be updated in the ontologies that represent this data. Second, ontologies are only useful when related to a particular application: they are not useful as an end product, but only as a tool to perform further tasks. They are also subjective and usually incomplete, so updates and modifications are often necessary. For example, an ontology may not contain all the different variants of a term (e.g. synonyms). Thus, ontology evolution is crucial in order to keep ontologies as up-to-date as possible, and to improve their relevance and usefulness.

This deliverable describes the continuation of work on various types of ontology evolution from metadata, as part of the ontology lifecycle. Previously, we have described methods for top-down ontology evolution, which incorporate a method for modelling some of the dynamics of (semantic) metadata, and which focused particularly on the interaction between the example clients and the NeOn toolkit, such that changes in networked ontologies can be propagated to the semantic metadata, and vice versa. From the bottom-up point of view, we have previously described preliminary versions of several systems that enable new information to be captured in ontological form and propagated to the NeOn toolkit, using data from unstructured text and folksonomies.

In this work, we describe improvements to all these systems, and also include some work on relational exploration, which is one of the most efficient approaches to semi-automatic ontology completion and refinement. These different approaches to ontology evolution can be divided into the following classifications:

- **Evolva, SPRAT and SARDINE** are all methods for **data-driven ontology evolution**. While Evolva makes use of information from unstructured data, other ontologies, and databases, SPRAT and SARDINE make use of information solely from unstructured data, but focus specifically on semantic annotations. We can call these systems **annotation-driven** as a specific kind of data-driven evolution. Evolva is described in Chapter 2, while SPRAT is described in Chapter 3[1].

- **RELExO** is a method for **hypothesis-driven ontology evolution**, which relies on several automatic components for suggesting possible refinements to the human ontology engineer. RELExO is described in more detail in Chapter 4.

[1]SARDINE is also mentioned in this chapter, but has been previously described in detail.

- **FLOR** (which was described in previous deliverables) is a method for **socially-driven ontology evolution**, since it exploits information from folksonomies in order to derive new semantic information.

These methods all make use of different techniques but complement each other, as they derive information from different sources in order to construct or refine ontologies, and can be synergetically combined in an integrated workflow. More specifically, suggestions for ontology changes generated by the data-driven and socially-driven evolution tools could be fed into RELExO, which computes the most likely hypotheses from the union of the individual result sets. These hypotheses along with provenance information, such as certainty values or source documents, are then presented to the human ontology engineer in order to support him in extending the ontology.

1.1 Related Deliverables

The work in this deliverable is strongly related to a number of other deliverables in NeOn:

1. It describes the final implementation of the methods for ontology change management proposed in D1.5.1, D1.5.2 and D1.5.3;
2. It is closely related to the work on change management to support collaborative workflows described in D1.3.2. The implementation of our change log system in GATE is designed specifically to interact with the change log and workflows implemented there.
3. The bottom-up approaches to ontology change management such as Evolva, SPRAT and SARDINE are closely related to work carried out in WP2 and WP7, and describe continued development from work in D1.5.2 and D1.5.3. SPRAT is described in more detail in D2.2.2.
4. Some later work on evaluating FLOR is reported in D2.2.4.
5. The development of RELExO is related to work carried out in WP3, and describes an extension of previous versions of the system described in D3.8.1 and D3.8.2.
6. The applications are all realised as NeOn Toolkit plugins described in WP6.

Chapter 2

Evolva

2.1 Introduction

Ontologies form the basis of Semantic Web systems. As such, they need to be kept up-to-date for the dependent systems to remain usable. A substantial part of ontology dynamics includes learning and evolving ontologies from external data sources, which provide potential sources of new domain knowledge to be represented in the ontology. Such ontology updates and changes mainly involve finding and selecting statements to be added to the ontology. Statement selection is sometimes based on conflict detection to keep the ontology coherent after adding such statements [HS05]. However, a conflict-free statement is not necessarily correct/relevant to an ontology. Even if extracted from a carefully selected data source, some statements might concern domains other than the one covered by the ontology, might not fit the conceptualization encoded in it, or might simply not add any value to the ontology. Current ontology evolution tools have paid little attention to the (semi-)automatic assessment of the relevance of statements to an ontology.

With the increase of complexity and changes occurring in the represented domains, ontology evolution becomes a painstaking and time-consuming process. We regard ontology evolution as the "timely adaptation of an ontology to the arisen changes and the consistent management of these changes" [HS05]. "Timely adaptation" suggests a quick adaptation, that can only be achieved by decreasing user involvement in the evolution process. However, most current approaches heavily rely on user input. Moreover, the definition suggests that a successful evolution can only be achieved by having both adaptation and change management. Yet no existing approach handles both tasks in one framework. One set of approaches considers evolution as the management of changes performed by users for preserving consistency [Kle04, NCLM06, Sto04, VPST05], while another set targets techniques for integrating new knowledge into the ontology [AHO06, BHSV06, NLH07, OGG07], without an extensive handling of change and evolution management.

Our hypothesis is that it should be possible to assess the relevance of a statement by analyzing the context in which it occurs, and by checking how the context complements the ontology to evolve. We use online ontologies to provide contexts where statements are used. We extract the context of a statement from an ontology as the sub-graph of the ontology surrounding the entities linked through the statement. We then use matching techniques to align such a context with the ontology to evolve, relating common entities and relations in both graphs. We have also developed a visualization tool that provides an overview of this mapping. Using such a visualization, we studied the intersection of the context of a statement with the ontology to evolve, which helped us identify clues and indications to support the assessment of the relevance of the statement to the ontology.

Applying this methodology leads to interesting results and raises a number of issues related to the use of the proposed approach. Indeed, looking at various examples, general patterns emerge from specific contexts that seem to indicate either the relevance or the non-relevance of the statement. Also, while the large variety of ontologies available on the Semantic Web provides a great source of contexts for this approach, it also leads to new research questions, concerning for example the use of contexts having different levels of granularity from the considered ontologies, or that apply different modeling principles.

In this chapter, we discuss our ontology evolution tool Evolva (Section 2.2), which evolves ontologies starting from external data sources. Evolva detects new ontological entities from such data sources, and links them through its relation discovery component, which relies on various sources of background knowledge. In addition to our evolution tool, we discuss the assessment of statements in terms of correctness and relevance, and their evaluation when applied to the fishery domain (Section 2.3).

2.2 Evolva Framework

Evolva is a comprehensive ontology evolution framework, covering a complete ontology evolution cycle including a) the performing of changes based on external data sources, and b) the management of these changes as shown in Figure 2.1.

We identify the need for evolution by contrasting the content of the ontology with that of external data sources. Such sources could be text documents, folksonomies, databases, or other ontologies. Each source requires a different method of content extraction handled by the *information discovery* component (first box in Figure 2.1).

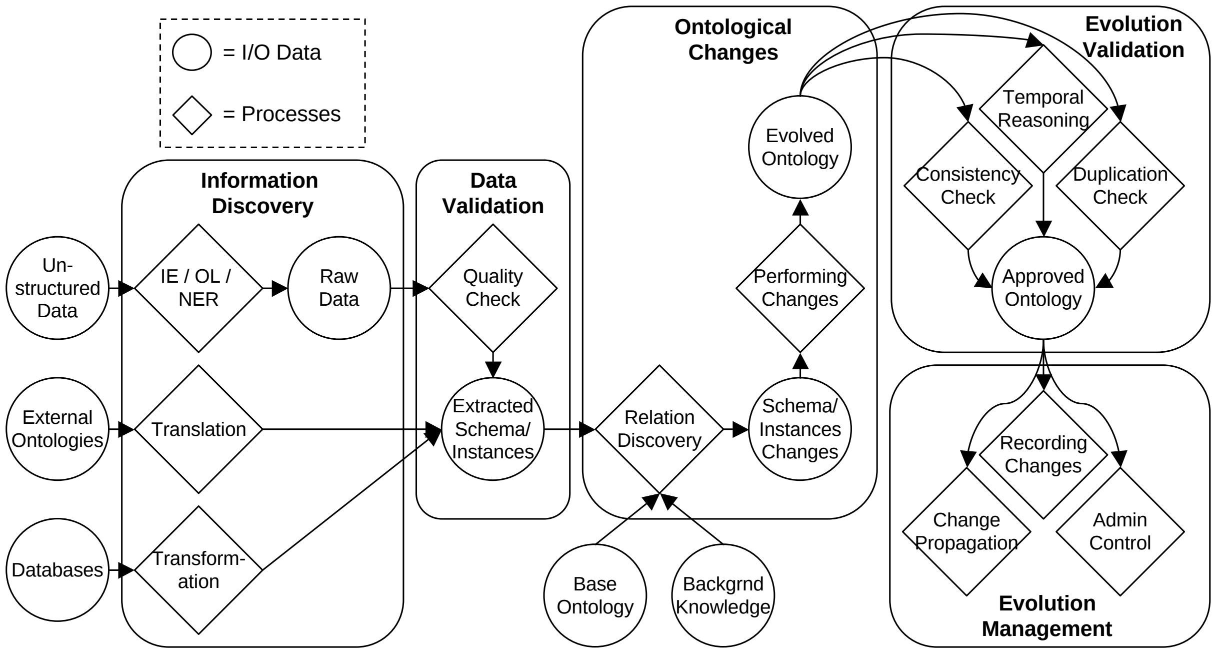


Figure 2.1: Evolva Framework Architecture

A detailed description of Evolva's components can be found in our previous NeOn deliverable D1.5.3 [MAP$^+$09]. In brief, the *data validation* component identifies new terms that are relevant for the ontology. It also checks the quality of content and filters out noise generated from the information discovery component. The validated information is passed to the *ontology changes* component in which background knowledge plays a core role in automating the integration of new information to the ontology. The use of background knowledge aims to decrease user input throughout the evolution. Background knowledge exists in various formats including lexical databases such as WordNet [Fel98], online ontologies, and unstructured web documents. Our relation discovery step uses background knowledge to determine the relationship path of new knowledge to existing knowledge in the ontology. The evolution could generate conflicts and problems that are handled at the level of the *evolution validation* component. We plan to have in this component temporal reasoning for time-related problems, coupled with duplication and consistency checks. Finally the validated ontology is passed to the *evolution management* component (bottom right part of Figure 2.1) where the user has control over the evolution, and changes are recorded and propagated to dependent ontologies.

Evolva has been implemented as a plugin for the NeOn toolkit. Currently it covers the data discovery from text documents, the relation discovery using WordNet and online ontologies as sources of background knowledge, in addition to integrating the changes on the ontology itself or on a separate new version. In addition, Evolva is fully compatible with the Change Capturing plugin, and all changes are recorded and integrated in the workflow for change management [PHYd08]. Implementation details are discussed in [MAP+09]. In this deliverable, we focus on improving the output of the *relation discovery* feature in Evolva, by providing statement validation techniques as presented in the following sections.

2.3 Statement Evaluation

Evolva automatically discovers new statements to be added to the ontology to increase its domain coverage, with respect to a set of external data sources. The question arises as to how such statements should be evaluated. Adding new statements to an ontology can have several effects on the ontology. Most current approaches limit their validation to conflict detection mechanisms, such that statements are only added if they do not cause conflicts in the ontology. However, adding an incorrect or irrelevant statement would not necessarily initiate an inconsistency state in the ontology. In order to obtain higher precision in the process of ontology evolution, it is worth evaluating the statements in terms of (1) correctness and (2) relevance with respect to the ontology.

2.3.1 Evaluating Statement Correctness

One of the assets of Evolva is in the use of online ontologies as background knowledge, as a source of new statements linking new entities to existing entities in the ontology. However, with little control over the quality of published ontologies, there remains a possibility of getting incorrect statements. The NeOn deliverable D2.2.3 [SdCd+09] describes in detail various techniques for evaluating ontological entities. The parts of D2.2.3 that mainly focus on evaluating the statement correctness are:

1. Trust-based Evaluation of Ontology Components: this technique relies on feedback from the community of users for assessing the quality of an ontology or ontological entity. Such feedback can be accessed through Watson [dBG+07], a Semantic Web gateway that collects, indexes and gives access to thousands of online ontologies, that can be searched and explored through its user interface, or through Web services. This would give a direct indication of the correctness of a statement, in order to decide whether or not to use it in the ontology evolution.

2. Other techniques are described in the deliverable for evaluating the correctness of ontology statements. Such methods are partly inspired from existing work in NLP and applied in Web/Semantic Web environments, and can be reused to evaluate the correctness of statements generated by ontology evolution tools.

2.3.2 Statement Correctness Experiment

We performed an experimental evaluation of the current implementation of the relation discovery module on the data sets provided by the KMi scenario. Our goal is to get an insight into the efficiency, in particular in terms of precision, of the relation discovery relying on our two main background knowledge sources: WordNet and online ontologies. Online ontologies are accessed using Scarlet [SdM08], a Semantic Web based relation discovery engine, which matches terms to online ontological entities for resolving relations [ES07].

Extracted Term	Ontology Concept	Relation	Relation Path
Contact	Person	$\sqsubseteq$	contact $\sqsubseteq$ representative $\sqsubseteq$ negotiator $\sqsubseteq$communicator$\sqsubseteq$ person
Business	Partnership	$\sqsubseteq$	business $\sqsubseteq$ partnership
Child	Person	$\sqsubseteq$	child $\sqsubseteq$ person

Table 2.1: Examples of relations derived using WordNet

	Evaluator 1	Evaluator 2	Evaluator 3	Agreed by all
Correct	106	137	132	76
False	96	53	73	26
Don't know	2	15	0	0
Precision	53 %	73 %	65 %	75 %

Table 2.2: Evaluation results for the relations derived from WordNet

Experimental Data

We used 20 documents from the KMi news repository[1] as a source of potentially new information, from which 520 terms were identified. The base ontology which we wish to evolve (i.e. KMi's ontology based on the AKT ontology) currently contains 256 concepts. By using the Jaro matcher we discovered that 21 of the extracted terms have exact correspondences within the base ontology and that 7 are closely related to some concepts (i.e. their similarity coefficient is above the threshold of 0.92).

Evaluation of the WordNet Based Relation Discovery

WordNet has been long used as a reference resource for establishing relations between two given concepts, based on the relation that exists between their synsets. Thanks to the WordNet based relation discovery module, 162 out of the 492 remaining new terms have been related to concepts in the ontology. Some of these relations were duplicates as they related the same pair of term and concept through the relation of different synsets. For evaluation purposes, we eliminated duplicate relations and obtained 413 distinct relations (see examples in Table 2.1).

We evaluated a sample of randomly selected 205 relations (i.e. half the total) in three parallel evaluations. This manual evaluation[2], which is not part of our evolution framework, helped to identify those relations which we considered correct or false, as well as those for which we could not decide on a correctness value ("Don't know"). Our results are shown in Table 2.2. We computed a precision value for each evaluator; however, because there was considerable variation between them, we decided to compute also a precision value on the sample on which they all agreed. Because of the rather high disagreement level between evaluators (more than 50%), we cannot draw a generally valid conclusion from these values. Nevertheless, they already give us an indication that, even in the worst case scenario, more than half the obtained relations would be correct. Moreover, this experiment helped us to identify typical incorrect relations that could be filtered out automatically.

[1] http://news.kmi.open.ac.uk
[2] To our knowledge, there are no benchmarks of similar experimental data against which our results could be compared.

Evaluation Results for Scarlet

Scarlet [SdM08] automatically selects and explores online ontologies *to discover relations between two given concepts*. For example, when relating two concepts labelled *Researcher* and *AcademicStaff*, Scarlet identifies (at run-time) online ontologies that can provide information about how these two concepts inter-relate, and then combines this information to infer their relation. [SdM08] describes two increasingly sophisticated strategies to identify and exploit online ontologies for relation discovery. We rely on the first strategy, which derives a relation between two concepts if this relation is defined within a single online ontology, e.g. stating that *Researcher* $\sqsubseteq$ *AcademicStaff*. Besides subsumption relations, Scarlet is also able to identify disjoint and named relations. All relations are obtained by using derivation rules which explore not only direct relations but also relations deduced by applying subsumption reasoning within a given ontology. For example, when matching two concepts labelled *Drinking Water* and *tap_water*, appropriate anchor terms are discovered in the TAP ontology and the following subsumption chain in the external ontology is used to deduce a subsumption relation: *DrinkingWater* $\sqsubseteq$ *FlatDrinkingWater* $\sqsubseteq$ *TapWater*. Note that, as in the case of WordNet, the derived relations are accompanied by a path of inferences that lead to them.

The Scarlet-based relation discovery processed the 327 terms for which no relation had been found in WordNet. It identified 786 relations of different types (subsumption, disjointness, named relations) for 68 of these terms (see some examples in Table 2.3). Some of these relations were duplicates, as the same relation can often be derived from several online ontologies. Duplicate elimination led to 478 distinct relations.

For the evaluation, we randomly selected 240 of the distinct relations (i.e. 50% of them). They were then evaluated in the same setting as the WordNet-based relations. Our results are shown in Table 2.4, where, as in the case of the WordNet-based relations, precision values were computed both individually and for the jointly agreed relations. These values were in the same ranges as for WordNet. One particular issue we faced here was the evaluation of the named relations. These proved difficult because the names of the relations did not always make their meanings clear. Different evaluators provided different interpretations for these, which increased the disagreement levels. Therefore, again we cannot provide a definitive conclusion of the performance of this particular algorithm. Nevertheless, each evaluator identified more correct than incorrect relations.

Summary

The overall precision of around 77% shows that background knowledge can largely contribute to automating the integration of new knowledge into the ontology. This is where user input is traditionally most needed. We also found that precision can be further increased through introducing validation techniques such as using the ontology itself as a relation validator, and by using filter mechanisms for excluding irrelevant terms.

No.	**Extracted Term**	**Ontology Concept**	**Relation**	**Relation Path**
1	Funding	Grant	$\sqsubseteq$	funding $\sqsubseteq$ grant
2	Region	Event	occurredIn	region $\sqsubseteq$ place $\longleftarrow$occurredIn- event
3	Hour	Duration	$\sqsubseteq$	hour $\sqsubseteq$ duration
4	Broker	Person	isOccupationOf	broker -isOccupationOf$\longrightarrow$ person
5	Lecturer	Book	editor	lecturer $\sqsubseteq$ academicStaff $\sqsubseteq$ employee $\sqsubseteq$ person$\longleftarrow$editor-book
6	Innovation	Event	$\sqsubseteq$	innovation $\sqsubseteq$ activity $\sqsubseteq$ event

Table 2.3: Examples of relations discovered using Scarlet

	Evaluator 1	Evaluator 2	Evaluator 3	Agreed by all
Correct	118	126	81	62
False	96	56	57	17
Don't know	11	47	102	8
Precision	56 $\%$	70 $\%$	59 $\%$	79 $\%$

Table 2.4: Evaluation results for the relations derived with Scarlet.

2.3.3 Methodology for Evaluating Statement Relevance

Evaluating the correctness of statements is only part of the process in reaching a higher quality ontology evolution. In many cases, there are a lot of new *correct* statements, which are discovered and proposed to enrich the base ontology. However, not all of them are useful to be added to the ontology. For example, if a user is evolving an ontology in the academic domain, and the concept *Death* occurs in a text document, Evolva would probably link *Death* as a type of *Event* in the ontology. Of course the statement by itself is correct and would not conflict with the ontology, but it is probably not relevant to add. With plenty of statements to check, it becomes a burden on the user to manually select the relevant ones. In this section, we present our methodology for identifying such relevance.

Exploring the Ontological Context of a Statement for Assessing its Relevance

One way to assess whether a statement is relevant to a particular ontology is to rely on additional information provided by background knowledge sources. More specifically, we consider that such background knowledge can be given by the contexts in which this statement has been used and applied. The main idea of our method is to explore the ontological contexts of statements, i.e. the contexts in which they are applied in other, external ontologies, to identify factors allowing to assess their relevance. In this work, it is assumed that the relevance of a statement relates to the added value of this statement with respect to a particular ontology. In other words, we want to check whether the considered statement should be added to the considered ontology.

To analyze the use of ontological contexts to assess the relevance of a statement s to an ontology O, three main tasks need to be realised. First, such ontological contexts have to be discovered and extracted. We use a relation discovery engine on the Semantic Web, based on a Semantic Web gateway to identify online ontologies where a statement s appears, and devise a technique to extract the surrounding of s in these ontologies. The result is a set of contexts $\{C_1, C_2, ..., C_n\}$ corresponding to sub-parts of ontologies surrounding s. Second, these ontological contexts have to be aligned to the ontology O. We use simple matching techniques to identify common entities in a context C_i and the ontology O. Then, the degree of overlap between the context of the statement and the ontology needs to be interpreted and translated into a relevance measure. However, this is not a trivial task. Therefore, in order to gain a better understanding of which elements of the mapping might indicate relevance, we performed a study of various cases of such mappings. For this, we visualise the mapping between C_i and O, and study their intersection and complementarity in the search for relevance factors. We have implemented a visualization tool that displays a merged graph based on this mapping, distinguishing clearly the common parts from the parts specific to the ontology and statement context.

Discovering and Extracting Statements' Contexts

In order to obtain ontological contexts in which a given statement appears, we also use the relation discovery engine Scarlet [SdM08]. Scarlet exploits Watson's Web services to find ontologies that (directly or indirectly) relate entities with each other. In our case, considering a statement s of the form $s =< Subject, relation, Object >$, we use Scarlet to find ontologies that relate the entities $subject$ and

$object$ through the relation $relation$.

Once the ontologies in which s appears have been identified, we extract the context surrounding s in each ontology. To achieve this, we traverse the existing relations between the entities $subject$ and $object$ linked by the statement s until a given recursion level is reached. This technique (which is very similar to the Prompt ontology view extraction feature [NM04] or some modularization techniques [dSSS09] using traversal approaches), includes a number of parameters to customize the size and the content included in the context. In our case, we use separate parameters to limit the extraction depth of the entities' *parents*, *children* and other *named relations*.

As an example, Figure 2.2 shows the result of extracting the context of the statement $< Project, has - funding, Grant >$ from an online ontology[3], with a recursion level of 1 for the *parents*, *children* and other *named relations*.

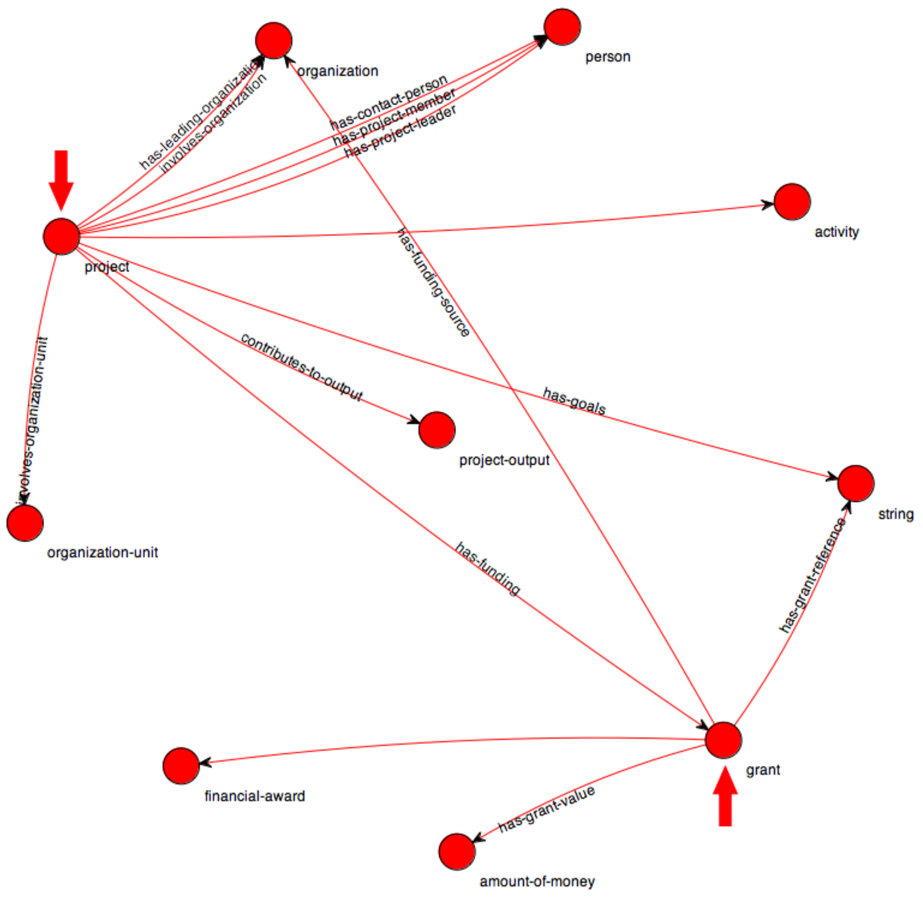


Figure 2.2: Example of ontological-context for the $< Project, has - funding, grant >$ statement

Context Graph Matching

The goal of matching the statement context to the ontology is to indicate how well the statement fits in the ontology. In other words, we want to analyse the intersection, as well as the differences, between the context of the statement s and the ontology O, to derive potential factors indicating how well the considered statement would fit in O. The method we apply relates to the task of ontology matching [ES07], but considers only a sub-part of one ontology (the context). The purpose of our method is essentially different from ontology matching: while ontology matching aims to assess how similar two knowledge structures are, our aim is to identify how complementary the two structures are (i.e. how the relation context completes or fits in the ontology to evolve).

[3]http://www.mindswap.org/2004/SSSW04/aktive-portal-ontology-latest.owl

The approach we use for matching can be seen as a graph matching process. Graphs here are the set of asserted statements in a context or ontology. It starts by matching the names of the nodes (i.e. the entities) in the graphs of C_i and O using the Jaro-Winkler string similarity [CRF03]. In a second step, it tries to align the edges of the graphs (i.e. the relations), first by comparing their names using the same similarity measure, and second by using the direction of the relation. We do not currently take into account the matching of relations in the analysis of our experiment.

Visualising Ontology-Statement Context Mappings

In order to analyze the mappings between statement contexts and the ontology to evolve, we developed a tool to visualise such a mapping, clearly showing the intersection as well as the differences between the two graphs. The mappings resulting from the process described above allow us to divide the elements of the context and the ontology into three groups: (1) entities that are common to both C_i and O, (2) entities that are only present in C_i and (3) entities that are only present in O. Our visualization displays a unique graph based on these three groups of elements, using different shapes and colours to distinguish them. Entities from the first group are merged and displayed in green, and represented as star-shaped nodes. Entities from the second group are represented in red, with round-shaped nodes. Entities from the third group are represented in blue, with square-shaped nodes.

Figure 2.3 shows an example of the mapping visualization between the SWRC ontology[4], and the context of the statement $< Deliverable, subClassOf, Report >$ in an online ontology describing bibTeX entries[5].

As can be seen from this example, depending on the size of the ontology and of the context, many elements might be displayed that are not useful to assess the relevance of a particular statement. Optional filters can then be applied to obtain a simpler visualisation, keeping only the surrounding concepts of the statement's nodes that are common in the two graphs (see Figure 2.4 for the same example with this filter applied), presenting a clearer view of the context match.

2.3.4 Relevance Patterns and Relevance Assessment Technique

The ultimate goal of our work is to automatically assess the relevance of a statement to an ontology based on the level of matching between its context and the ontology (thereby exploiting the result of the techniques presented above). We conducted a study of which factors could be taken into account for computing relevance. Concretely, in the study presented in this section, we applied the techniques above to obtain contexts for 10 statements and then assess the relevance with respect to the SWRC ontology. We chose this set of statements because it contained both relevant and irrelevant ones. Table 2.5 shows some examples of the selected statements.

We report here on the observations we derived from analysing the visualization of the corresponding mappings presented above in our methodology. We discuss two relevance patterns we detected in these examples, and an additional third pattern generated as a side effect of Pattern 1. These patterns correspond to general situations appearing in the mapping graphs and indicating that the statement is relevant.

Pattern 1: adding a sub-class to a joint concept (non-leaf in the ontology to evolve)

Pattern 1 occurs when the statement to be assessed includes a concept being a *subClassOf* a concept occurring in both the context and the ontology (see the example in Figure 2.4, where the considered statement is $< Deliverable, subClassOf, Report >$ and the shared concept is $Report$). The factor indicating relevance in this case is the fact that other subclasses of the common concept are also shared between the context and the ontology, are close to each other (like $TechReport$ and $TechnicalReport$ in our case) or are somehow related to each other. Such a pattern clearly indicates the relevance of the statement, as it

[4]http://ontoware.org/frs/download.php/354/swrc_updated_v0.7.1.owl
[5]http://oaei.ontologymatching.org/2004/Contest/103/onto.rdf

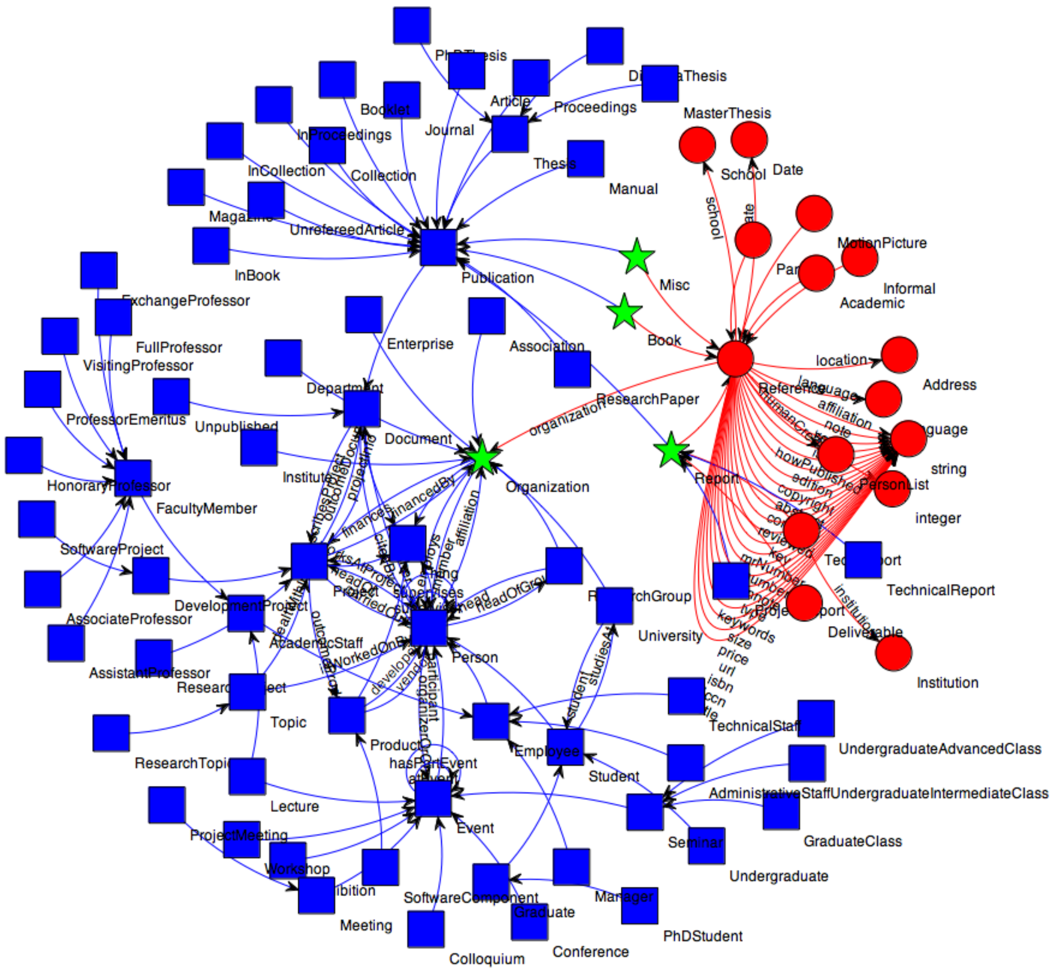


Figure 2.3: Example of visualizing the mapping

shows that, while the context and the ontology have many subclasses of the same concept in common, the one to be added through the statement is missing in the ontology. Thus we calculate the degree of confidence of Pattern 1 by taking the ratio of the new concept's siblings that are in common between the context of the statement and the ontology to evolve, to the total number of the new concept's siblings according to the following formula:

$$Conf(p1) = \frac{|Siblings(NewC) \cap SubClassOnt(CommonC)|}{|Siblings(NewC)|}$$

where $p1$ refers to Pattern 1, $NewC$ to the new concept suggested to be added to the ontology, $CommonC$ to the concept in common between the ontology to evolve and the context, and $SubClassOnt$ to a function for extracting the subclasses of C in the ontology to evolve.

Conversely, if the conditions of the pattern do not hold for a particular context mapping, it also provides an indication of the irrelevance of the statement. Such a counter example is presented in Figure 2.5, where the statement considered is $< Player, subClassOf, Person >$. In this case, the concept to be added ($Player$) and its siblings are not related to the subclasses of $Person$ in the SWRC ontology.

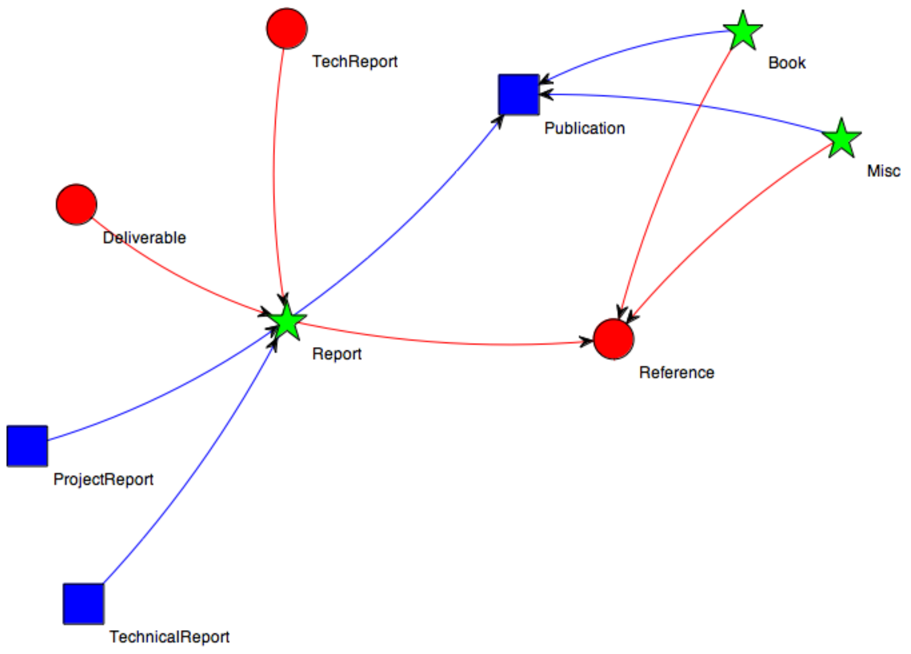


Figure 2.4: Simplified visualization of the mapping

Pattern 2: adding a super-class to a joint concept

Pattern 2 occurs when the statement considered would add a superclass to a concept common to the context and the ontology. In this case, similarly, if the other subclasses of the added concept also occur in both ontologies or are somehow related, this represents an indication of the relevance of the statement. For example, this pattern appears with the statement $< Organization, subClassOf, ActingEntity >$ in Figure 2.5 where the joint concept $Person$ is also a subclass of $ActingEntity$, therefore reinforcing its addition to the ontology.

We calculate the confidence of Pattern 2 based on the ratio of subclasses in common between the statement context and the total number of subclasses in the context, as in the following formula:

$$Conf(p2) = \frac{|SubClassContext(NewC) \cap ClassList(Ont)|}{|SubClassCont(NewC)|}$$

Where $SubClassContext$ is a function to retrieve subclasses of a concept in its context and $ClassList$ is a function to get all classes in an ontology.

Figure 2.4 provides another interesting case of Pattern 2 for $< Book, subClassOf, Reference >$. The addition of this statement to the ontology is reinforced by the fact that $Reference$ could be the superclass of two joint concepts. However, since $Reference$ and $Publication$ denote similar concepts, adding $Reference$ would be redundant and would add little value to the ontology. In contrast, $ActingEntity$ is sufficiently different from other concepts in the ontology to be a worthwhile addition.

Pattern 3: adding a subclass to a joint concept, being a leaf in the ontology to evolve

This is a variation of the initial Pattern 1, which fails if the common concept is a leaf in the ontology to evolve (as there will be no subclasses in this case). We spotted this pattern during our user evaluation, where many relevant statements have been undetected before the introduction of Pattern 3. This pattern is especially useful in scenarios where ontology extension to more specific concepts is needed.

Statement	Context Ontology	Relevant	Figure
$< Project, has-funding, Grant >$	http://www.mind\discretionary{-}{}{}swap.org/2004/SSSW04/ aktive-portal-ontology-latest.owl	Yes	2.2
$< Deliverable, subClassOf, Report >$	http://ontoware.org/frs/download.php/ 354/swrc_updated_v0.7.1.owl	Yes	2.3
$< Organization, subClassOf, ActingEntity >$	http://www.atl.external.lmco.com/projects/ ontology/ontologies/basketball_soccer/ basketball.daml	Yes	2.5
$< Player, subClassOf, Person >$	http://www.atl.external.lmco.com/projects/ ontology/ontologies/basketball_soccer/ basketball.daml	No	2.5
$< Media, disjoint, Event >$	http://watson.kmi.open.ac.uk/ontologies/ LT4eL/CSnCSv0.01Lex.owl	No	N/A

Table 2.5: Example of Statements with Relevance with respect to the SWRC Ontology

The indication of relevance in Pattern 3 is when the context of the statement has classes in common with the ontology to evolve. The more common elements there are, the more relevant the statement is. To get a better indication of common elements, we focus on the part of the evolving ontology where the statement is most likely to end up, by extracting a subgraph of the corresponding ontology around the common concept. This is very useful in the case where the ontology to evolve is of considerable size. We use the following formula for the Pattern 3 formula calculation:

$$Conf(p3) = \frac{|ClassList(Context) \cap ClassList(OntGraph)|}{|ClassList(Context)|}$$

Where $OntGraph$ is the subgraph of the ontology starting from the common node, up to a customisable depth (distance between the common node and the leaf of the subgraph generated around the common node of the ontology).

2.3.5 Statement Relevance Experiment

In addition to the statement correctness experiment, we performed an experiment to evaluate our statement relevance assessment approach. The goal of the experiment was to get an insight into how well our pattern-based algorithm would perform, and to identify potential new patterns to introduce in our algorithm.

Experimental Data

We performed our experiment in the fishery domain, taking the BioSphere[6] ontology as the ontology to evolve, with a fishery related corpus extracted from the (www.fishonline.org) website composed of 109 text documents at the time of experiment. The task here is to evolve the ontology by focusing on adding new knowledge about the fishery domain. We used Evolva to identify potentially new concepts from the corpus, and online ontologies as the source of background knowledge.

Evolva identified 216 new statements to be added to the ontology. We filtered out statements with highly generic concepts (e.g. Fish $\sqsubseteq$ *Individual* or Bird $\sqsubseteq$ *Object*) which are clearly irrelevant to add. We also focused on the *subClass* relations, leaving the named relations' relevance as part of our future work. We ended up with 124 statements, out of which we randomly selected a set of 100 statements for evaluation.

[6]http://kmi-web06.open.ac.uk:8081/cupboard/ontology/Experiment1/biosphere?rdf

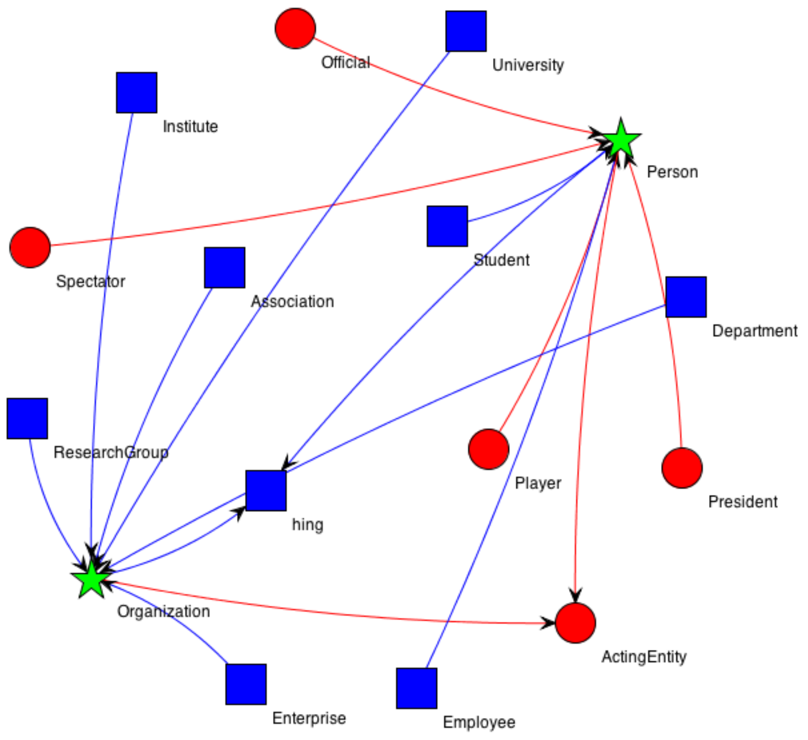


Figure 2.5: Pattern 2 example for $< Organization, subClassOf, ActingEntity >$. Pattern 1 counter example for $< Player, subClassOf, Person >$.

Relevance Evaluation Results

Our data set of 100 statements has been evaluated by three users using a customized Web interface (shown in Figure 2.6). A statement can be evaluated *Relevant* if it is important to add to the ontology, *Irrelevant* if it is better to discard it, or *Don't Know* if the user cannot judge from the given information. In the case of irrelevance, the user has the option to specify if the irrelevance is due to the fact that the statement is incorrect, or if it does not fit in the ontology.

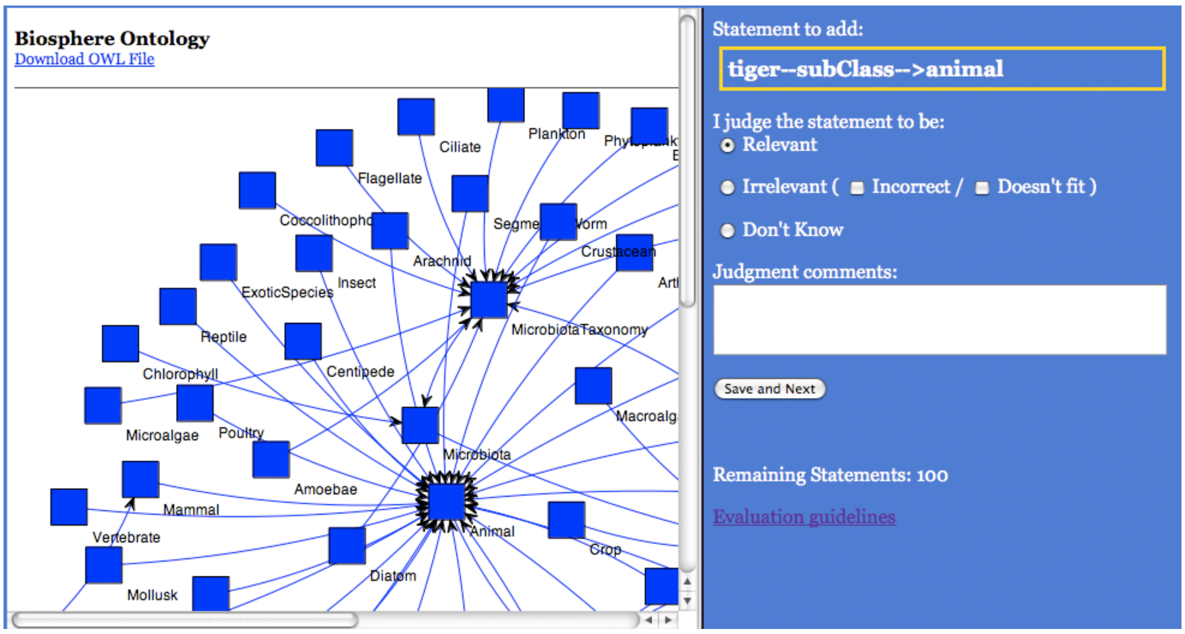
Given the three possible answers by users, we represent each answer by a value "Relevant" = 1, "Don't Know"=0.5, and "Irrelevant"=0. Then we take the average of the answers per statement as a representation of an overall relevance evaluation:

$$Rel(s) = Ans(u1) + Ans(u2) + Ans(u3)$$

where Rel is the overall relevance returned by the users, s is the corresponding statement, and Ans is the answer of the user.

Then we take two thresholds: a $relevance\_threshold$, $irrelevance\_threshold$ between the range [0,3]. As shown in Figure 2.7, a statement s is $relevant$ $iff(Rel(s) > relevance\_threshold)$, while it is $irrelevant$ $iff(Rel(s) < irrelevance\_threshold)$, or $undetermined$ $iff(irrelevance\_threshold < Rel(s)$ < $irrelevance\_threshold)$.

We have also assigned a threshold per type of pattern to evaluate the corresponding confidence value. In this experiment, we set the threshold values as shown in Table 2.6. The evaluation results are presented in Table 2.7. We calculate the precision/recall of our algorithm based on the formula below. Our algorithm was able to find 30 of the 36 relevant statements judged by the users, giving 83% recall. In terms of irrelevant statements, the recall is 63%. The 84% precision in finding irrelevant relations is acceptable, however we



Statement to add:

tiger--subClass-->animal

I judge the statement to be:

[✓ Relevant]

Irrelevant (Incorrect / Doesn't fit)

Don't Know

Judgment comments:

Remaining Statements: 100

Figure 2.6: Statement relevance evaluation interface.

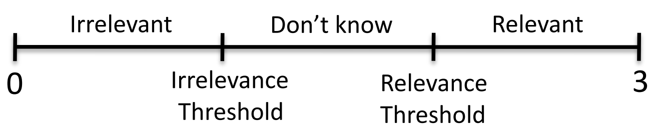


Figure 2.7: Statement relevance set by threshold.

will be working on improving our algorithm to increase the 53% precision in spotting the relevant statements. Overall, in this tested domain scenario, our algorithm is considered to behave well in filtering out irrelevant statements, and finding most of the relevant ones.

$$Precision = \frac{User\_AlgorithmAgreement}{TotalRelevancebyAlgorithm}$$

$$Recall = \frac{User\_AlgorithmAgreement}{TotalRelevancebyUser}$$

where $User\_AlgorithmAgreement$ is the total of relevance agreed between the users and our algorithm.

2.4 Conclusions

Ontology evolution is a tedious and time consuming task, especially at the level of introducing new knowledge to the ontology. Most current ontology evolution approaches rely on the ontology curator's expertise to come

Threshold	**Value**
Relevance	2
Irrelevance	1
Pattern 1	0.6
Pattern 2	0.1
Pattern 3	0.02

Table 2.6: Threshold values.

Fishery Dataset	**Users**	**Algorithm**	**User_Algorithm Agreement**	**Recall**	**Precision**
Relevant (2-3)	36	56	30	83.33%	53.57%
Don't Know (1-2)	6	N/A	N/A	N/A	N/A
Irrelevant (0-1)	58	44	37	63.79%	84.09%

Table 2.7: Evaluation results for relevance assessment

up with the right integration decisions. We have discussed in this chapter how background knowledge can support Evolva, our ontology evolution framework, for automating the process of relation discovery. In our experiments, we explored WordNet and Semantic Web ontologies (through the Scarlet relation discovery engine).

Being able to assess the correctness as well as relevance of a statement to be added to an ontology is crucial in ontology evolution as well as for other tasks such as ontology learning or enrichment. However, such a task is generally left to the ontology developer, as it requires background knowledge to understand the added value of the statement to the considered ontology.

In this chapter, we have envisaged an approach based on the use of the context in which a statement is used in external ontologies, to assess its relevance with respect to the ontology to evolve. We have presented a methodology to study the relation between this ontology and ontological contexts extracted from online ontologies, with the goal of obtaining relevance factors. Applying this methodology to several concrete examples taken from the Evolva scenario, we were indeed able to extract relevance patterns from the graph-based visualization employed to study the mapping between the ontology and the contexts in which the considered statements appear.

Our experiments show the feasibility of using background knowledge in ontology evolution with an average precision of 77%. We also report in this chapter an experiment on assessing the relevance of statements with respect to ontologies under evolution. While our initial results seem very promising so far, we shall continue our work in refining our patterns, and finding additional patterns to cover a better relevance detection.

Chapter 3

Annotation-driven Ontology Learning

3.1 Introduction

In this section, we describe the final implementation of the SPRAT and SARDINE applications, which form part of the GATE Web Services plugin[1]. These applications are also detailed in [MFP09b] and [MFP09a]. These exemplify the idea of annotation-driven ontology learning, which can be seen as a particular kind of data-driven ontology learning. As discussed in previous deliverables D1.5.2 and D1.5.3, these applications represent a bottom-up approach to ontology evolution, starting from the text and resulting in a modified ontology. On the other hand, we have also previously described a top-down approach to ontology evolution, in the form of a change management system which ensures that changes made to an ontology are reflected at the annotation level and can be applied to other parts of a networked ontology. We describe here the finalisation of the change management system developed in GATE in order to ensure compatibility between changes made to an ontology in GATE and in the NeOn toolkit. This ensures that one can edit ontologies in either architecture while retaining compatibility. We have applied the change log mechanism in GATE automatically to the SPRAT and SARDINE applications, so that when an ontology is modified using these applications, a change log is stored in GATE and can be applied to the original version of the ontology in the NeOn toolkit (or in GATE) or viewed separately in either system. This is necessary because the GATE web service plugins are only loosely coupled, so when changes are made to the ontology, compatibility would not necessarily be ensured and changes would not necessarily be made available to other users. The combination of the change management system in GATE, the existing change management system in the NeOn toolkit, and the GATE web service plugins thereby creates a complete ontology development cycle.

In the following section, we describe the implementation of SARDINE and SPRAT. The two applications both generate new ontology data from text: the difference is that SARDINE is tuned specifically to the fisheries domain and makes use of the FAO species ontology as a seed ontology, while SPRAT is a generic version that can be run on any kind of text and does not make use of a seed ontology. SARDINE was developed specially for the FAO use case in collaboration with WP7, while SPRAT is aimed at more general use.

3.2 SARDINE and SPRAT

In this section, we describe our work on SARDINE and SPRAT, which generate potential new concepts for the ontology, based on analysis of the text. This work is carried out in conjunction with work in WP2 and WP7. In D2.2.2 [VTSFGP$^+$08], we described the initial work extracting entities from unstructured text and thereby reengineering information from textual documents into ontologies.

SARDINE aims to find new mentions of fish and other marine life from a corpus, and adds them to the ontology as new instances or concepts (or makes suggestions about where to add them). It operates as a GATE Annotation Service (GaS), taking as input the text to be processed and producing as output an OWL

[1]http://gate.ac.uk/projects/neon/webservices-plugin.html

file. SPRAT aims to find new mentions of any kind of concept or instance from the text, and add them to the ontology in the right place. It also may find synonyms and some other properties of existing instances. Since both applications are so similar, and since SARDINE has already been described in some detail in D1.5.2, we shall describe SPRAT in more detail and then explain how SARDINE differs.

3.2.1 GATE application

SPRAT (Semantic Pattern Recognition and Annotation Tool) is composed of a number of GATE components: some linguistic pre-processing followed by a set of gazetteer lists and the JAPE transducers (grammars) described above. The components are as follows:

- Tokeniser: divides the text into tokens
- Sentence Splitter: divides the text into sentences
- POS-Tagger: adds part-of-speech information to tokens
- Morphological Analyser: adds morphological information (root, lemma etc.) to tokens
- NP chunker: divides the text into noun phrase chunks
- Gazetteers: looks up various items in lists
- OntoRootGazetteer (optional): looks up items from the ontology and matches them with the text, based on root forms
- JAPE transducers: annotate text and add new items to the ontology

The application can either create an ontology from scratch, or modify an existing ontology. The ontology must be loaded with the application (in the former case, a blank ontology is loaded; in the latter, the ontology to be modified) and referenced by the grammar via the runtime parameter. The ontology used is the same one for the whole corpus: this means that if a number of documents are to be processed, the same ontology will be modified. If this is not the desired behaviour, then there are two options:

1. A separate corpus is created for each document or group of documents corresponding to a single output ontology. The application must be run separately for each corpus.
2. A processing resource can be added to the application that clears the ontology before re-running on the next document. This of course requires that the ontology is saved at the end of the application, after processing each document.

In the web service version of the application, the default behaviour is the only one possible, however (i.e. that a single ontology is incrementally enriched by the processing of each document in the corpus).

3.2.2 Lexico-syntactic patterns for ontology generation

In D2.2.2, we described in some detail the set of lexico-syntactic patterns used in SPRAT and SARDINE, so we shall simply provide a summary here, and discuss some extensions to the patterns we have developed. We have identified three sets of patterns which can help us identify concepts, instances and properties to extend the ontology: the well-known Hearst patterns [Hea92], the Lexico-Syntactic Patterns developed in the NeOn project corresponding to Ontology Design Patterns [PGS$^+$08, dCGPPSF08] and some new contextual patterns defined by us [MFP09b, MFP09c].

In D2.2.2, we performed some preliminary evaluation of the SPRAT application, and drew a number of conclusions about its performance and suggestions for future work:

- incorporation of deeper semantic relations using semantic classes from VerbNet and WordNet in order to look for verbal patterns connecting terms in a sentence.
- investigating the use of TermRaider for restricting the number of candidates for extraction. TermRaider[2] is a term extraction tool we developed in-house, which is also available in its own right as part of the GATE webservice plugin for the NeOn Toolkit.
- incorporate combinations of Hearst patterns and statistically derived collocational information.

Currently we have implemented the first two of these suggestions. First, we applied restrictions on the patterns, based on incorporating deeper semantic relations using semantic classes from VerbNet[Sch05] and WordNet[Fel98] in order to look for verbal patterns connecting terms in a sentence, and to restrict the kinds of noun phrase extracted. We made use of the ANNIC plugin in GATE [ATBC05] to search for frequently occurring annotation patterns. We aim not only to reduce the number of errors, but also to eliminate the kind of general relations which while not incorrect, are not very useful. For example, knowing that a turtle is a local creature is not of much interest unless more contextual information is provided (i.e. in which region it is local).

We took inspiration also from some currently unpublished research carried out at DFKI in the Musing project[3], which looks at deriving T-Box Relations from unstructured texts in German. In this work, attention is focused primarily on deriving relations between parts of German compound nouns, but we can make use of similar restrictions. For example, in their work they might derive from the compound noun "bank manager" that there is a property "has manager" belonging to "bank", and that a "bank manager" is a subclass of "manager".

3.2.3 Restrictions on Noun Phrases

We prevent certain stop words occurring as part of a noun phrase recognised in the patterns. These stop words are a combination of some words given the wrong grammatical category by the part of speech tagger, and some common modifiers which should not be included in the noun phrase. For example, words such as "baby", "adult" (when modifying an NP) and most adjectives of size, number etc. should not be included. This prevents things like "baby elephant" being recognised as a subclass of "elephant". Adjectives of colour, on the other hand, may often be used to denote subclasses: for example, a "white rhino" is a kind of rhino. This list of stop words was determined heuristically and can be augmented as necessary with further iterations of testing.

3.2.4 Restrictions on Subclass Patterns

We modified the subclass rule `(Adj|N) NP<class>` $\rightarrow$ `NP<subclass>` from the set of contextual patterns, such that either the superclass must already exist in the ontology as a recognised class, or such that certain semantic restrictions apply. For example, one restriction states that both the proposed subclass and superclass must have the semantic category "animal". This enables us to recognise relations such as "carrot weevil" as a subclass of "weevil". This rule in particular has very high accuracy (98%) and only seems to cause errors as a result of incorrect semantic categories from WordNet.

3.2.5 Restrictions on Properties

One of the most error-prone rules was the Property rule `X has Y` from the Lexico-Syntactic Patterns set, which was clearly far too general. We restricted this to again use semantic categories of WordNet. For example, for animals we can state that X must be an animal and Y must be a body part. This gave much better results (75% accuracy, although low recall). Another restriction is the type of thing that can be considered

[2] http://gate.ac.uk/projects/neon/termraider.html
[3] http://www.musing.eu

Relation	Total Extracted	Correct	Precision
Subclass	163	79	48.5%
Instance	21	10	47.6%
Synonym	98	47	48.0%
Property	107	24	22.4%

Table 3.1: Results of relation extraction on 25 wikipedia documents

Relation	Total Extracted	Correct	Precision
Class	1058	884	83.6%
Subclass	659	505	76.6%
Instance	23	12	52.2%
Synonym	98	47	48.0%
Property	55	41	74.5%

Table 3.2: Results of relation extraction after refinement

a property. We experimented with restricting the range of the property to the following semantic categories from WordNet: plant, shape, food, substance, object, body, animal, possession, phenomenon, artifact, and found much improved results.

3.2.6 Restrictions on Concepts

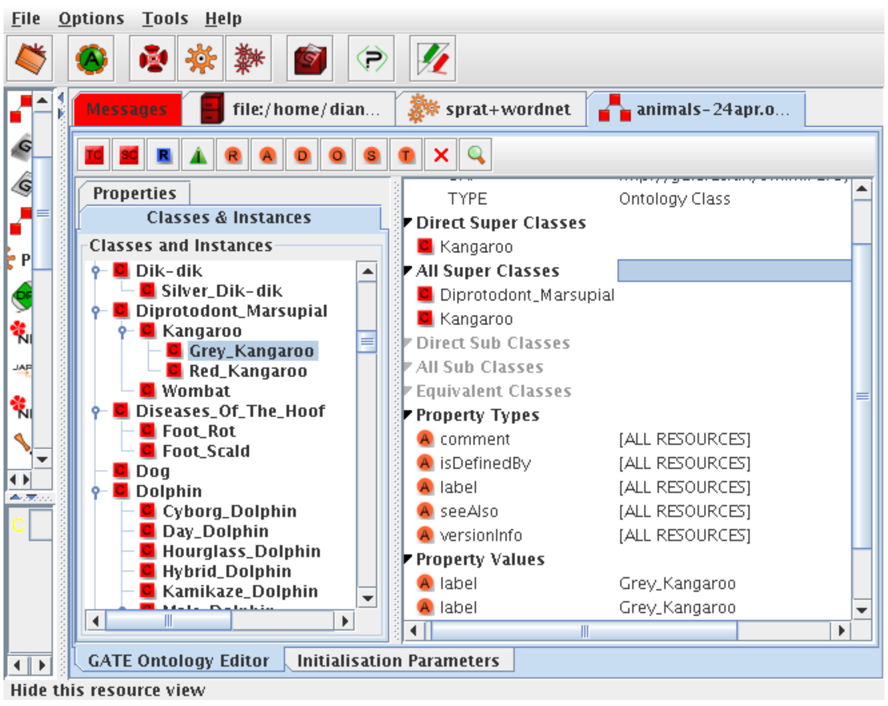
As proposed, we investigated the use of TermRaider in order to limit items proposed as new concepts to those which had been first identified as terms. This helped greatly in preventing some more common words being proposed as new concepts, which while correct, were not really relevant for the ontology.

The results obtained after the restrictions were applied were very encouraging. Table 3.1 shows our previous results, reported in D2.2.2, while Table 3.2 shows the results after the improvements were made. Note that the figures are higher in the improved version due to incorporating TermRaider – most items recognised by TermRaider were included as new classes, which also led to recognition of more subclasses.

SPRAT generated 1058 classes, of which 83.6% were correct; 659 subclasses, of which 76.6% were correct, 23 instances, of which 52.2% were correct, and 55 properties, of which 74.5%

3.2.7 Implementation of patterns

The patterns are implemented in GATE as JAPE rules. JAPE is a pattern matching language developed by the University of Sheffield [CMT00] and used extensively in GATE applications. On the left hand side (LHS) of the rule is the pattern to be annotated. This consists of a number of pre-existing annotations which have been created as a result of pre-processing components (such as POS tagging, gazetteer lookup and so on) and earlier JAPE rules. The implementation of the lexico-syntactic patterns was described in detail in D2.2.2. The right hand side (RHS) of the rule invokes NEBOnE and creates the new items in the ontology, as well as adding annotations to the document itself. This part of the rule first gets the relevant information from the annotations (using the labels assignFigureed on the LHS of the rule), then adds the new information to the ontology and finally adds annotations to the entities in the document. NEBOnE is responsible also for ensuring that the resulting changes to the ontology are wellformed: this was described in more detail in D2.2.2. Figure 3.1 shows a screenshot from GATE of an ontology created from a document about sharks.



Classes and Instances

- Dik-dik
  - Silver_Dik-dik
- Diprotodont_Marsupial
  - Kangaroo
    - Grey_Kangaroo
    - Red_Kangaroo
  - Wombat
- Diseases_Of_The_Hoof
  - Foot_Rot
  - Foot_Scald
- Dog
- Dolphin
  - Cyborg_Dolphin
  - Day_Dolphin
  - Hourglass_Dolphin
  - Hybrid_Dolphin
  - Kamikaze_Dolphin

```
TYPE                     Ontology Class
Direct Super Classes
  Kangaroo
All Super Classes
  Diprotodont_Marsupial
  Kangaroo
Direct Sub Classes
All Sub Classes
Equivalent Classes
Property Types
  comment                [ALL RESOURCES]
  isDefinedBy            [ALL RESOURCES]
  label                  [ALL RESOURCES]
  seeAlso                [ALL RESOURCES]
  versionInfo            [ALL RESOURCES]
Property Values
  label                  Grey_Kangaroo
  label                  Grey_Kangaroo
```

Figure 3.1: Generated ontology in GATE

3.3 GATE Change Log

In D1.5.3 [MAP+09] we described our work on developing a change log management system within the GATE Ontology API such that it is compatible with the one produced by the NeOn toolkit. Although some basic testing was carried out to check if the change logs were compatible with NTK, we could not test them thoroughly because the NTK did not at that time have a functionality to load and apply change logs to existing ontologies. Since then, a new functionality has been added in the NTK which enables the production of new change logs and the loading of existing ones. In this deliverable, we explain how the system was modified in order to enable this final stage of compatibility.

There is a difference between manually verifying change logs and having a functionality in NTK to test these change logs. The obvious difference is the time it takes to verify change logs and therefore the number and the variety of tests that can be undertaken. With the latest development in the NTK, it becomes easier to test the change logs by loading them in the NTK and checking by simply browsing through the ontology. We found the following issues when applying the GATE change logs in the NTK:

1. Modifications to the change log since the last version:

   We observed that there were some minor changes in the NTK change log since the last version of the NTK. The reference change logs used for our first implementation were outdated and therefore we had to use the latest change logs to make appropriate changes to the GATE software.

2. No support for owl:import statements:

   Changes recorded in change logs are recorded as instances of the classes that represent these changes. For example, an instance of AddClass is created to record an addition of a class in the ontology. Similarly an instance of RemoveClass is created to record deletion of a class. The taxonomy which defines these classes is stored in a separate file. In the earlier version, we referred to such ontologies by simply importing them in the change log. However, after experimentation, it was discovered that the NTK did not support owl:import statements. We therefore had to make changes to the GATE change log management system to include the definition of the classes/properties used in the change log.

3. Different conventions for URI formation:

   As mentioned earlier, changes are recorded as instances of the classes that represent these changes. The NTK has different conventions for forming different types of URIs. For example, it uses the following convention to form a URI to represent a change:

   ```
   <defaultNameSpace>?location=&<typeOfInstance>=<randomCharacterSequence>
   ```

   Here, the typeOfInstance could be "axiom" or "change". It turned out that the conventions used by GATE were not compatible with the conventions used in NTK. Instead, we only used the random character sequence that followed the default name space. Although the difference in conventions did not prevent the loading of change logs, to make them identical and easier for debug, we changed our system to follow the new conventions set by the NTK.

4. Difference in atomic changes:

   An entity change can give rise to more than one atomic change. For example, in the NTK adding an instance is a two step process: an instance is declared and then the class assertion is made. In GATE, however, no declaration of individuals is required. This resulted in only one change being recorded for the addition of individuals. We fixed this problem with an explicit mention of these changes, and at the same time interpreting them correctly while applying change logs produced by the NTK inside GATE.

5. Missing entries in the change specification log:

   It was observed that the entity changes were not recorded correctly in the history log. The NTK creates an individual of the Log concept with a property hasLastChange, pointing to the last change made to the ontology. While parsing a change log, the NTK looks at this instance and traverses backwards to obtain the previous changes. In order to do so, it looks at the value of hasPreviousChange property set on every instance of change. It was observed that in some cases, GATE did not set the value of this property, which resulted in incorrect traversal. This issue was resolved.

Having dealt with the above issues, we were able to produce change logs from GATE and apply them in NTK successfully.

3.4 Conclusions

In this chapter, we have presented the final implementation of the SPRAT and SARDINE applications, which include the ability to record change logs and to apply these in the NeOn toolkit to previous versions of an ontology. It should be noted, however, that while the applications are intended for use in real contexts (such as the FAO fishery use case in WP7), they are very much prototype applications and could be extended and improved. Work on the applications does not therefore stop here, and continued improvements will be made, in particular to SPRAT. We have only tested a small number of restrictions to the lexico-syntactic patterns, and this will form part of future development, along with increasing the number of patterns. This work will be carried out in collaboration with work by other partners on the development of Ontology Design Patterns.

Chapter 4

RELExO

4.1 Introduction

Relational exploration is one of the most efficient approaches to semi-automatic ontology completion and refinement. The underlying attribute exploration algorithm as well as the involvement of a description logic reasoner help to reduce the required user effort to a minimum. However, a certain degree of user interaction seems unavoidable. The research and development efforts we are reporting in this chapter can be considered a first step towards fully automatic relational exploration.

In this chapter we describe an extended version of RELExO, our tool for semi-automatic ontology refinement based on Formal Concept Analysis (see NeOn D3.8.1 [VB08]). Unlike the original release, it features an additional automatic expert which uses textual background knowledge for suggesting possible answers to the human ontology engineer. Even though the implementation of this expert is still ongoing, we report preliminary results and give an elaborate example illustrating the added value that could be provided by such an automatic expert.

4.2 Relational Exploration with Automatic Experts

4.2.1 Relational Exploration

Figure 4.1 shows the conceptual architecture of RELExO, our framework for reasoner-aided relational exploration, which consists of an exploration component and a team of automatic and human experts. The icons on the left side of the diagram symbolize the two experts (*thinker* and *document*) as well as the KAON2 reasoner (*gear wheels*). In the course of the exploration, all of them are asked for the validity of hypotheses $C \sqsubseteq D$ and the existence of counterexamples γ. Depending on their respective answers, RELExO updates the ontology KB_{Σ}, the formal context or the implication base.

The ontology management infrastructure of KAON2[1] is used for querying and updating the ontology, while two experts, a human ontology engineer and an automatic (e.g. ontology learning) expert, validate the hypotheses brought up by the exploration component as depicted by Figure 4.2. Each *hypothesis*, corresponding to a potentially missing subsumption axiom $C \sqsubseteq D$ with complex or atomic classes C and D, can be accepted (i.e. confirmed) or rejected by the experts. If the hypothesis is accepted, RELExO adds the corresponding axiom to the ontology KB_{Σ}. If an expert rejects a hypothesis, they have to provide a *counterexample*, that is an individual γ which is a member of C and *not* a member of D. In case the hypothesis is a disjointness axiom $C_1 \sqcap C_2 \sqsubseteq D$, for example, this must be an individual which instantiates the left-hand side of the subsumption, $C_1 \sqcap C_2$, as every individual is per definition *not* a member of the empty class $\bot$. Each counterexample is as well added to the ontology, before the exploration continues by computing the next hypothesis. For further details regarding the implementation of RELExO and a sample runthrough,

[1]`http://kaon2.semanticweb.org`

please refer to [VR08b] and [VR08a].

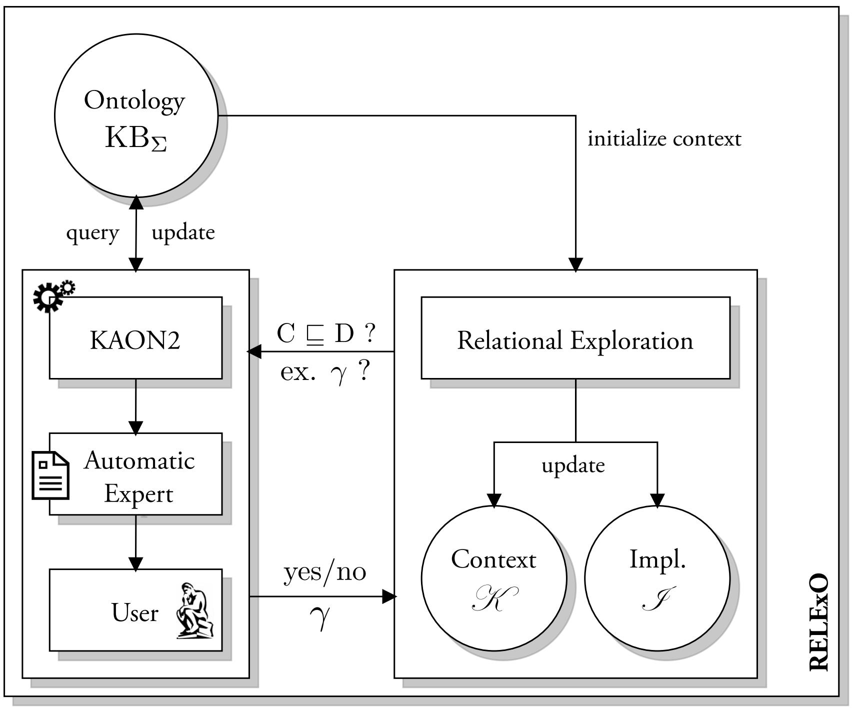


Figure 4.1: RELExO's conceptual architecture

4.2.2 The Wikipedia Expert

As illustrated by Figure 4.1, the Wikipedia-based automatic expert extends the "expert team" consisting of the KAON2 reasoner, which could be considered some kind of logical expert, and the human ontology engineer. It is always called prior to the human expert and provides him (or her) with suggestions for possible answers. This is also illustrated in Figure 4.2, where the *gear wheels* represent the KAON2 reasoner, while the *thinker* symbol indicates user involvement. Whenever the non-logical, Wikipedia-based expert is called, this is denoted by a *document* symbol. The implementation of the automatic non-logical expert is based on association rules which are computed from Wikipedia (see also further below). An *association rule* is an implication $X \rightarrow Y$ with *itemsets* $X, Y \subseteq T$ and $X \cap Y = \emptyset$.

For an association rule $X \rightarrow Y$ and a set of items $T = t_1, ...t_n$ *support* and *confidence* are defined as follows:

$$\text{support}(X \rightarrow Y) = \frac{|\{t \in T | X \cup Y \ \subseteq \ t\}|}{|T|}$$

$$\text{confidence}(X \rightarrow Y) = \frac{|\{t \in T | X \cup Y \ \subseteq \ t\}|}{|\{t \in T | X \ \subseteq \ t\}|}$$

Like Mädche and Staab [MS00], we transfer this definition into a text mining setting. For this purpose, we build a corpus $D = d_1, ...d_n$ of encyclopedic documents, each of them describing a concept $c \in C$ as indicated by the title of the document. We can then assume each item set $t_i \in T$ to consist of those concepts $c_{i,1}, ...c_{i,m} \in C$ which are mentioned by document d_i.

Example. {Zoo, Whale} $\rightarrow$ {Aquarium} is an association rule which states that the co-occurrence of the words "Zoo" and "Whale" implies that "Aquarium" is contained in the same document. Given a corpus of ten documents – eight of them mention both "Zoo" and "Whale" and six among those eight include all of the words (i.e. "Zoo", "Whale" and "Aquarium") – we would obtain a support value of $\frac{6}{10} = 0.6$ and a confidence value of $\frac{6}{8} = 0.75$.

The automatic expert translates each hypothesis $C \subseteq D$ into an association rule, where each atomic class corresponds to a search term. The support and confidence values can then be seen as indicators for the likeliness of the conclusion terms to occur in a Wikipedia articles, given that it contains all the terms in the premise. An article which contains all the terms in the premise and lacks many of the conclusion terms is considered a potential counterexample.

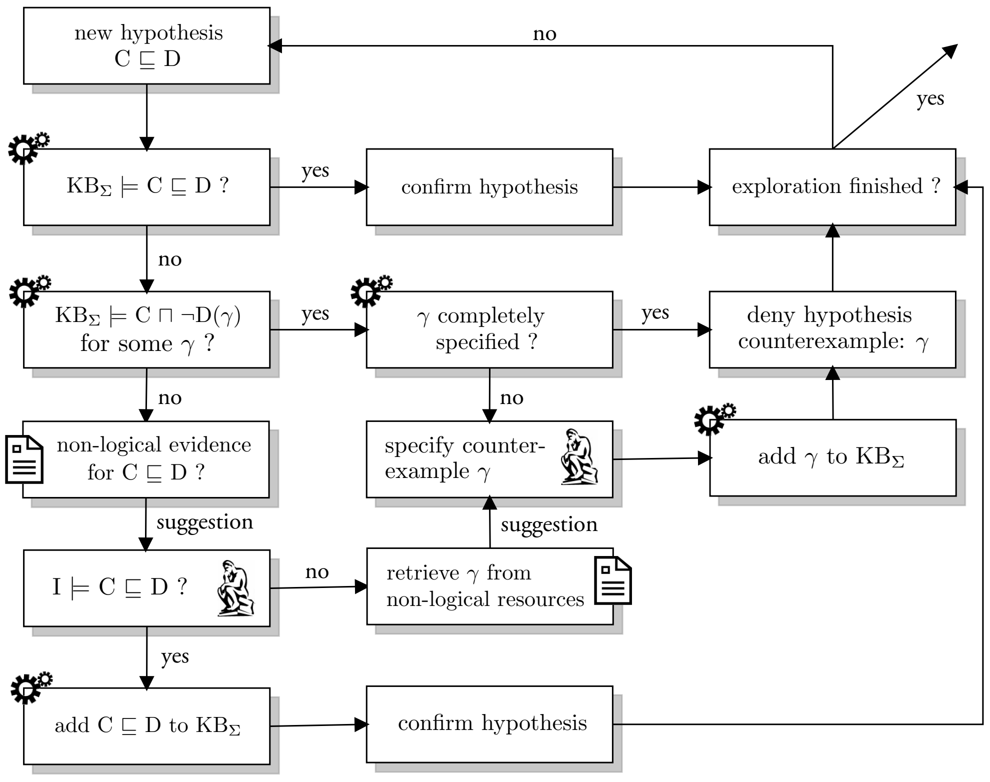


Figure 4.2: Relational exploration process

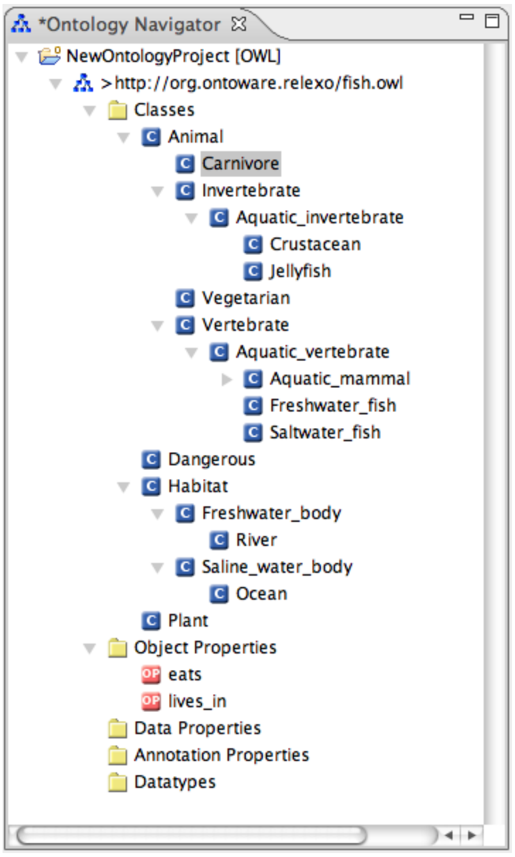
In the following sections, we describe two example scenarios which illustrate the use of automatic experts in a relational exploration setting.

4.3 Scenario I: Knowledge Acquisition

Since the current prototype of the automatic non-logical expert has been finished only recently and a detailed evaluation would have delayed the finalization of this deliverable, we will just give a short example (i.e. a complete run of the existing implementation) which illustrates the existing functionality and leave more extensive evaluation experiments for future work. Nevertheless, even though this example is not based on *real* case study data, we are confident that a systematic user study will demonstrate the usefulness of our approach when it comes to the refinement and evaluation of the FAO ontologies (WP7). This is why we decided to perform initial tests with data from the fishery domain.

4.3.1 Data Set

The data set we used for testing our implementation consists of a reasonably large text corpus and a small, inexpressive ontology of the fishery domain. The corpus has been constructed by retrieving all the 4,709 Wikipedia articles which are contained in the "Fish" category and its subcategories. It has an overall size of about 30 Megabytes. The fishery ontology, which we constructed ourselves in an adhoc manner, is much smaller consisting of only 19 atomic classes. A screenshot of the ontology's taxonomic hierarchy is shown by Figure 4.3.



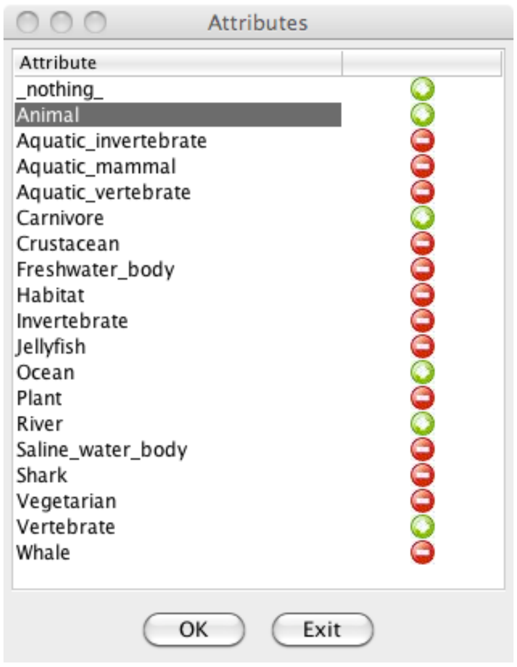
```
*Ontology Navigator
NewOntologyProject [OWL]
  >http://org.ontoware.relexo/fish.owl
    Classes
      Animal
        Carnivore
        Invertebrate
          Aquatic_invertebrate
            Crustacean
            Jellyfish
        Vegetarian
        Vertebrate
          Aquatic_vertebrate
            Aquatic_mammal
            Freshwater_fish
            Saltwater_fish
      Dangerous
      Habitat
        Freshwater_body
          River
        Saline_water_body
          Ocean
      Plant
    Object Properties
      eats
      lives_in
    Data Properties
    Annotation Properties
    Datatypes
```

Figure 4.3: Manually built ontology about fish

4.3.2 Exploration

Before the user can start exploring the ontology, he has to specify the focus of the exploration by selecting a set A of atomic classes (cf. Figure 4.4). In this diagram, as indicated by the *plus* symbols, the user has already selected the attributes "Animal", "Carnivore", "Vertebrate", "Ocean" and "River". This dialogue is displayed at the beginning of each relational exploration session, i.e. before the first hypothesis is raised by RELExO. In the current implementation of RELExO, any two descriptions C and D which are part of a hypothetical axiom $C \sqsubseteq D$ must only consist of conjunctions of classes contained in the previously specified set A.

As usual the exploration starts by proposing the hypothetical axiom $\top \sqsubseteq \bot$ (cf. Figure 4.5). In this diagram, the upper part of the dialog depicts the premise, i.e. the left-hand side of the subsumption axiom, and the lower part represents the right-hand side. This dialog is displayed each time RELExO comes up with a new hypothesis that needs to be validated by the human expert. First, KAON2 is called to check whether this hypothesis is true or not. The reasoner does not find any evidence for the corresponding axiom to be entailed



Attribute	
nothing	+
Animal	+
Aquatic_invertebrate	−
Aquatic_mammal	−
Aquatic_vertebrate	−
Carnivore	+
Crustacean	−
Freshwater_body	−
Habitat	−
Invertebrate	−
Jellyfish	−
Ocean	+
Plant	−
River	+
Saline_water_body	−
Shark	−
Vegetarian	−
Vertebrate	+
Whale	−

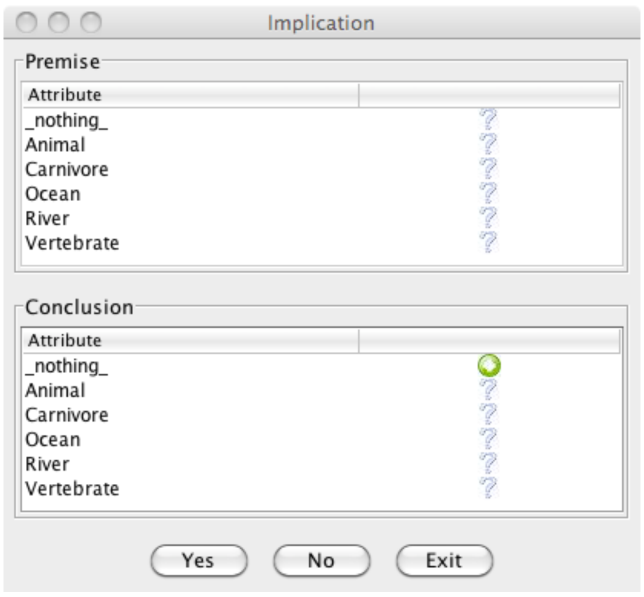
Figure 4.4: Attributes (i.e. atomic classes) representing the focus of the exploration

by the ontology. However, since the ABox does not contain any individuals at this time, there is nothing which might serve as a counterexample. So, the reasoner can neither prove nor disprove the correctness of the hypothesis and the other experts – the automatic Wikipedia-based expert and the human ontology engineer – need to be involved. Both of them reject the hypothesis as the set of domain entities would otherwise be empty and thus have to provide a justification in the form of a counterexample.

While the user immediately enters “Shark” as a counterexample to this rather trivial hypothesis (see Figure 4.6), the automatic expert tries to retrieve an appropriate counterexample from the Wikipedia corpus. It does so by constructing a query which contains all the attributes from the premise and as many negated attributes from the conclusion as possible. Whenever a hypothetical axiom gets rejected, this dialogue is shown to the user. A plus indicates that the individual (“Shark”) has a certain attribute, e.g. being a carnivore, whereas a minus denotes the fact that it does not have this attribute. In this example, sharks are modelled as carnivorous vertebrate animals which are neither rivers nor oceans. Note that the user has to specify only those attributes of an individual which make it a valid counterexample for the given hypothesis. An unspecified attribute would be indicated by a question mark. In this particular case, the set of attributes in the premise is empty and *all* the attributes in the conclusion are negated, as the attribute `nothing` is marked with a ’+’. The fulltext index of the corpus answers the query by returning the following Wikipedia article that seems to be an optimal counterexample insofar as it contains all the keywords:

> “*A bottom feeder is an aquatic animal that feeds on or near the bottom of a body of water. The body of water could be the ocean, a lake, a river, or an aquarium. ’Bottom feeder’ is a general term which is used particularly in the context of aquariums. More specific terms for bottom feeders are: groundfish, demersal fish and benthos.* (...)” [`http://en.wikipedia.org/wiki/Bottom_feeder`]

The second hypothesis brought up by RELExO is $\top \sqsubseteq Animal \sqcap Carnivore \sqcap Vertebrate$. Like the first one, it is rejected by both the automatic and the human expert. In order to find an appropriate counterexample, the former searches for Wikipedia articles which do *not* contain any of the attributes in the conclusion (i.e. “Animal”, “Carnivore” and “Vertebrate”) while mentioning at least some of the other attributes, that is


Implication

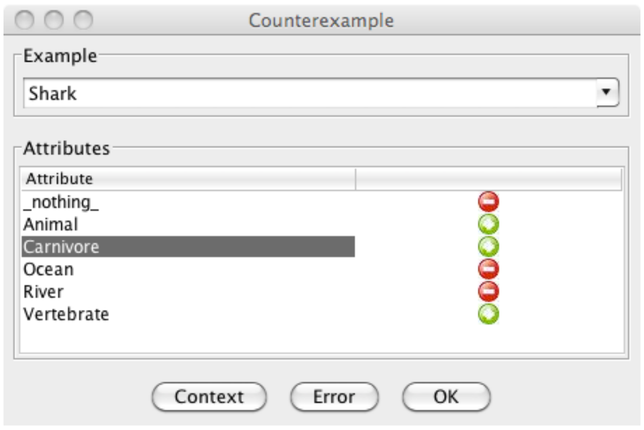
Premise

Attribute	
nothing	?
Animal	?
Carnivore	?
Ocean	?
River	?
Vertebrate	?

Conclusion

Attribute	
nothing	✓
Animal	?
Carnivore	?
Ocean	?
River	?
Vertebrate	?

Figure 4.5: Hypothesis $\top \sqsubseteq \bot$


Counterexample

Example: Shark

Attributes

Attribute	
nothing	⛔
Animal	✓
Carnivore	✓
Ocean	⛔
River	⛔
Vertebrate	✓

Figure 4.6: Counterexample

"Ocean" and "River". These criteria are met by an article about the Two Oceans Aquarium in South Africa – a reasonable counterexample, even though it does not instantiate any of the atomic classes or attributes, respectively. However, the user decides to ignore this suggestion and enters "AtlanticOcean".

"*The Two Oceans Aquarium is an aquarium located at the Victoria & Alfred Waterfront in Cape Town, Western Cape, South Africa. The aquarium opened on the 13 November 1995 and comprises seven exhibition galleries with large viewing windows.* (...)" [http://en.wikipedia.org/wiki/Two_Oceans_Aquarium]

Obviously, the next hypothetical axiom ($Animal \sqsubseteq Carnivore \sqcap Vertebrate$) is invalid, too. After computing the support and the confidence for the association rule {"Carnivore"} $\rightarrow$ {"Animal", "Vertebrate"} and comparing them to predefined empirically determined thresholds,[2] the automatic expert rejects the hypothesis. The human agrees and gives the answer "No". Searching for potential counterexamples the Wikipedia-based

[2]Both the support value of 0.00166 and the confidence value of 0.00441 are lower than the respective threshold.

expert now tries to find documents containing “Animal”, but *not* “Carnivore” or *not* “Vertebrate” and finally retrieves the article further below. Even though the article (“Ethics of eating meat”) matches the search criteria, it does not represent a useful counterexample, so the human overrides this suggestion by giving the counterexample “Squid”.

“*In most societies, controversy and debate have arisen over the ethics of eating animals. Ethical objections are generally divided into opposition to the act of killing in general, and opposition to certain agricultural practices surrounding the production of meat.* (...)” [`http://en.wikipedia.org/wiki/Ethics_of_eating_meat`]

The fourth hypothesis ($Animal \sqsubseteq Carnivoret$), which states that every animal must be a carnivore, is also rejected by the user – following the suggestion of the automatic expert who has computed a support value of 0.01464 and a confidence value of 0.03888. In order to find a potential counterexample, the automatic expert searches for an article that contains “Animal”, but *not* “Carnivore” and finally suggests “Animal testing” to the user. Since this answer is considered nonsense by the ontology engineer, he decides to give another counterexample, namely “Herring” with the attributes “Animal” and “Vertebrate”.

“*Animal testing, also known as animal experimentation, animal research, and in vivo testing, is the use of non-human animals in experiments. It is estimated that 50 to 100 million vertebrate animals worldwide – from zebrafish to non-human primates – are used annually.* (...)” [`http://en.wikipedia.org/wiki/Animal_testting`]

In this line, the exploration continues. The hypothetical subsumption $Animal \sqcap Ocean \sqsubseteq \bot$ (“`Animal` and `Ocean` are disjoint.”) is accepted by the two experts, and RELExO adds it to the ontology.

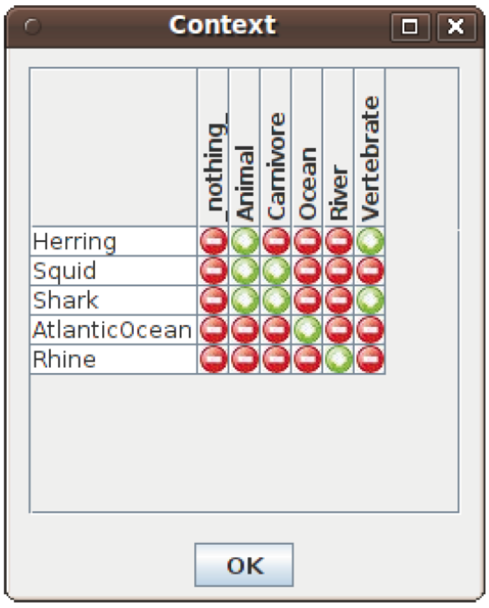
The hypothesis $River \sqsubseteq \bot$ which states the emptiness of the class `River` is rejected by the user, and the automatic expert give the same answer as the support value of 0.20721 exceeds the threshold. It suggests “Blackwater river” as a counterexample – a plausible answer, but as the corresponding Wikipedia article describes a class rather than an individual, the human ontology engineer decides to enter “Rhine”.

“*A blackwater river is a river with a deep, slow-moving channel that flows through forested swamps and wetlands. As vegetation decays in the water, tannins are leached out, resulting in transparent, acidic water that is darkly stained, resembling tea or coffee. Most major blackwater rivers are in the Amazon River system and the Southern United States.* (...)” [`http://en.wikipedia.org/wiki/Blackwater_river`]

The seventh axiom suggested by RELExO is $Animal \sqcap River \sqsubseteq \bot$ and hence states that nothing can be both an animal and a river. Since this is obviously correct, the user answers “Yes”, whereas the automatic expert rejects the hypothesis due to the support value of 0.11002.

Finally, the disjointness of “Ocean” and “River” ($Ocean \sqcap River \sqsubseteq \bot$) is confirmed by both experts and the exploration process terminates.

After all, the ontology contains 3 new axioms – all of them disjointness axioms – and 5 individuals, which were added as counterexamples (see Figure 4.7). The *plus* symbols indicate an incidence relation, i.e. the fact that an individual belongs to a certain class, whereas a *minus* means that the individual is *not* a member of that class. Although in this relatively simple scenario the acquired axioms could have easily been added manually, it is obvious that this kind of automatic support will become indispensable in case of bigger ontologies, where the number of potentially missing axioms is much higher.



	nothing	Animal	Carnivore	Ocean	River	Vertebrate
Herring	✗	✓	✗	✗	✗	✓
Squid	✗	✓	✓	✗	✗	✗
Shark	✗	✓	✓	✗	✗	✓
AtlanticOcean	✗	✗	✗	✓	✗	✗
Rhine	✗	✗	✗	✗	✓	✗

Figure 4.7: Formal Context

4.4 Scenario II: Knowledge Fusion

Various approaches for fully automated ontology enrichment have been proposed, of which some have been discussed earlier in this deliverable. Automatic methods have the advantage of requiring very little or no user interaction. This advantage, however, also leads to the main disadvantage: enrichments generated using these methods may be erroneous and usually will contain some mistakes. RELExO can result in very accurate results, but requires more user interaction. In this second scenario, RELExO is extended to support controlled enrichment of an ontology based on (possibly erroneous) enrichments automatically generated using other methods.

The extension consists of an additional logical expert, which on the one hand provides the standard reasoning tasks of a logical expert as described above, and on the other hand provides provenance information which can be used to import selectively from other ontologies and to further guide exploration. This additional service is based on *meta knowledge reasoning* as described in D3.1.4.

4.4.1 Meta Knowledge

The following definition of meta knowledge and related reasoning tasks is an excerpt from [QS09]. For a detailed definition, we refer the reader to [QS09].

When exploiting explicit or inferred knowledge in the semantic web, one must not only handle the knowledge itself, but also characterizations of this knowledge, e.g. (i) where a knowledge item came from (provenance); (ii) what level of trust can be assigned to a knowledge item; (iii) what degree of certainty is associated with it. We refer to all such kinds of characterizations as *meta knowledge*. In semantic web applications, meta knowledge needs to be computed along with each reasoning task.

Meta knowledge can come in various complex dimensions. Many simplifications performed currently, such as assuming trust to be measured on a scale from 1 to 10, are not justified. In contrast, actual information sources, modification dates, etc. should be tracked to establish trust [Har09]. We propose a flexible mechanism for tracking meta knowledge, which meets these requirements.

Meta knowledge can have multiple dimensions, e.g. uncertainty, a least recently modified date or a trust metric. For this work, we assume that these (and possible further) dimensions are independent of each other.

Definition 1 *Knowledge dimension. A knowledge dimension D is an algebraic structure $(B_D, \vee_D, \wedge_D)$, such that $(B_D, \vee_D)$ and $(B_D, \wedge_D)$ are complete semi-lattices.*

B_D represents the values the meta knowledge can take, e.g. all valid dates for the least recently modified date or a set of knowledge sources for provenance. As $(B_D, \vee_D)$ and $(B_D, \wedge_D)$ are *complete* semi-lattices, they are, in fact, also lattices. Hence, there are minimal elements in the corresponding orders.

Axioms can be assigned meta knowledge from any of the meta knowledge dimensions. Within a single assignment, the meta knowledge must be uniquely defined. Basically, a meta knowledge assignment is a list of attribute value pairs assigned to an axiom, such that the attribute is a meta knowledge dimension and the value is a value from this dimension.

Definition 2 *Meta Knowledge Assignment.*
A meta knowledge assignment M is a set $\{(D_1, d_1 \in D_1), ..., (D_n, d_n \in D_n)\}$ of pairs of meta knowledge dimensions and corresponding truth values, such that $D_i = D_j \Rightarrow d_i = d_j$.

Without loss of generality, we assume a fixed number of meta knowledge dimensions. As a default value for D_n in a meta knowledge assignment, we choose $\perp_D$.

To allow for reasoning with meta knowledge, we need to formalize how meta knowledge assignments are combined. *How provenance* [GKT07] is a strategy which describes how an axiom A can be inferred from a set of axioms $\{A_1, ..., A_n\}$, i.e. it is a boolean formula connecting the A_i. We call a logical formula expressing how provenance a *meta knowledge formula*. For example, the following query finds all limbs that are either broken or wrenched:

x : Limb $\wedge$ ($\langle x$, true$\rangle$: isBroken $\vee$ $\langle x$, true$\rangle$: isWrenched).

The results of this query and the corresponding meta knowledge formulae are:

limb1 $|\ \#_1 \wedge \#_3$ and limb2 $|\ \#_2 \wedge \#_4$

The operators for meta knowledge dimensions extend to meta knowledge assignments, allowing us to compute meta knowledge for entailed knowledge by evaluating the corresponding meta knowledge formula. Here, we extend operations on atomic meta knowledge values to assignments, i.e. we define operations on lists of attribute value pairs.

Definition 3 *Operations on Meta Knowledge Assignments.*
Let A, B be axioms and meta$(A) = \{(D_1, x_1), ..., (D_n, x_n)\}$ and meta$(B) = \{(E_1, y_1), ..., (E_m, y_m)\}$ be meta knowledge assignments. Let dim(A) be the set of meta knowledge dimensions of A. Then meta$(A) \vee$ meta$(B) = \{(D, x \vee_D y) | (D, x) \in$ meta(A) and $(D, y) \in$ meta$(B)\}$. $\wedge$ is defined analogously.

Having defined the operations on meta knowledge assignments, we can define formulae using these operations.

Definition 4 *Meta Knowledge Formula.*
Let A be an axiom of an ontology O, lab *a function assigning a unique label to each A_i from O and* lab(O) *the set of all labels of axioms in O. A meta knowledge formula ϕ for a axiom A wrt. an ontology O is boolean formula over the set of labels $\{lab(A_1), ..., lab(A_n)\}$ of axioms $\{A_1, ..., A_n\}$ from O, such that for each valuation $V \subset lab(O)$, which makes ϕ true, the following holds: $lab^-(V) \models A$.*

The meta knowledge of an axiom A within a meta knowledge dimension is obtained by evaluating the corresponding meta knowledge formula after replacing axiom labels with the corresponding meta knowledge in the dimension under consideration.

Definition 5 *Meta Knowledge of an Axiom.*
Let meta *be a function mapping from an axiom to a meta knowledge assignment in dimension D. The meta knowledge of an axiom A wrt. O in D is obtained by evaluating the formula obtained from A's meta knowledge formula wrt. O by replacing each $lab(A_i)$ with the corresponding $meta(A_i)$.*

4.4.2 The Meta Knowledge Expert

A dataset used by the Meta Knowledge Expert consists of

- an ontology to be enriched;
- multiple other auxiliary ontologies, which may have been automatically enriched;
- mappings from the ontology to be enriched to the auxiliary ontologies.

The mappings again may have been learned automatically. All learned axioms are annotated with confidence degrees generated by the learning algorithm.

We associate ontology axioms with meta knowledge through axiom annotations. Basically, an axiom annotation assigns an annotation object to an axiom e.g. "(brokenLimb subClass Limb) was created by Crow on 15.01.2008". A meta knowledge annotation consists of an annotation URI and a meta knowledge object specifying the value of the annotation. In our case, the meta knowledge object is a constant-value representing who asserted/modified the axiom, when the axiom was last modified, or the uncertainty degree of the axiom, the data source the axiom comes from, or a combination thereof. The grammar for meta knowledge annotations as an extension of OWL 2 annotations[3] is as follows:

OWLAxiomAnnotation := 'OWLAxiomAnnotation'
'('**OWLAxiom OWLAnnotation**$^+$')'
OWLAnnotation := **OWLConstantAnnotation**
OWLConstantAnnotation := **MetaKnowledgeAnnotation**
MetaKnowledgeAnnotation := 'MetaKnowledgeAnnotation'
'('**AnnotationURI MetaKnowledge**$^+$')'
MetaKnowledge := **CertaintyAnnotation** | **DateAnnotation** | **SourceAnnotation** | **AgentAnnotation**
CertaintyAnnotation := 'CertaintyAnnotation'
'('**AnnotationValue**')'
SourceAnnotation := 'SourceAnnotation' '('**AnnotationValue**')'
DateAnnotation := 'DateAnnotation' '('**AnnotationValue**')'
AgentAnnotation := 'AgentAnnotation' '('**AnnotationValue**')'

As the different ontology enrichment approaches discussed in this deliverable use different scales for confidence degrees, they need to be normalized first. Normalization may also take into account preferences between data sources or mappings. These preferences could be modelled as separate meta knowledge dimensions. For the data used in the experiments of the automatic methods, detailed trust and provenance metadata is not available. As only confidence degrees are available, only these are used. Using multidimensional meta knowledge is left for future work. It can be based for example on modification data derived from the history of Wikipedia articles used as input.

The meta knowledge agent implements the same functionality as the logical agent (i.e. KAON2) described in section 4.2, with one difference: while for the generation of hypotheses only the ontology to be enriched is used, confirmations and counterexamples are computed using the whole ontology network. This means that the automatically computed but potentially erroneous enrichments from earlier enrichment steps are used to improve the guided exploration. This reduces the number of questions necessary to ask the user.

[3] OWL 2 Web Ontology Language: Spec. and Func.-Style Syntax: http://www.w3.org/TR/2008/WD-owl2-syntax-20081202

For each confirmation and counterexample respectively, the meta knowledge degrees - in our case confidence degrees - are computed. Results whose confidence degrees fall below a user specified threshold are discarded. A higher threshold means more automatically learned results are discarded, and hence more questions need to be asked to the user. In contrast, a low threshold means fewer questions, but a higher probability of introducing errors. If the threshold is set to 0, the meta knowledge agent behaves just like a 'normal' logical agent using the whole ontology network, such as the logical agent described in Section 4.3.

If a confirmation or counterexample has been found, the explanation with the highest confidence degree for this result is returned and can be used to further help the user choose whether to accept or discard this enrichment. The explanation with the highest confidence degree is not necessary the same as the most relevant one as defined in [JQH08]. While the latter aims at returning the most comprehensive explanation, the meta knowledge agent returns the one with the highest confidence degree.

4.5 Future Work

We have limited the discussion in the previous section to rather primitive confidence degrees, which do not make full use of the capabilities of the meta knowledge framework. In the future we would like to take into account more detailed information of the agent creating knowledge fragments, modification dates and trust assignments. This will require both enriched datasets and modelling of preferences among data sources. Moreover, detailed explanations of proposed enrichments can make it easier for the user to decide whether certain enrichments should be made. This will require a corresponding extension of the user interface and an evaluation of the most efficient guidance to the user.

The implementation of the automatic expert described in Section 4.2 and 4.3 can be further improved by methods for bridging the lexical gap between the corpus and the queries. These methods could be based on Latent Semantic Indexing or machine processable dictionaries such as WordNet, in order to obtain evidence for lexical semantic relations between terms (e.g. synonymy, antonymy or hyponymy).

Furthermore, a full replacement of the human ontology engineer as suggested in the introduction will demand more sophisticated error handling approaches, including uncertainty and inconsistency management. For this purpose, RELExO must be enabled to automatically resolve logical contradictions which result from incorrect answers of automatic experts [Ser08]. We will also consider implementing a variant of formal concept analysis suitable for dealing with fuzzy or probabilistic contexts.

Finally, we would like to develop similar automatic experts for the exploration of other logical fragments such as generalized domain-range restrictions. These could be integrated, for example, into RoLExO, our tool for the semi-automatic refinement of property restrictions (see NeOn D3.8.2 [VB09]).

4.6 Conclusion

The development of non-logical automatic experts for relational exploration is a difficult task which, to the best of our knowledge, has not been attempted before. Our prototypical implementation based on association rules and text mining techniques yields very promising results, but needs to be improved significantly before it can be used in a real-world setting. As none of the approaches used provide perfect results, we have proposed knowledge fusion based on meta knowledge reasoning in order to support the user in combining results from multiple learning approaches into a single high quality one. In order to be able to use a fully automatic approach, sophisticated error handling and additional dimension of meta knowledge will be needed. In a broader context, it will be an interesting research question to evaluate use and usefulness of automatically generated, mapped and enriched ontologies on the Web. A lot of Linked Data, a very recent movement, is based on such automatically generated data. Even though it contains a certain degree of errors, it is extremely useful due to its sheer size and tight integration, which would not have been possible using manual methods alone. Hence, automatic ontology enrichment will remain an interesting area of research in the future.

Chapter 5

Conclusion

In this deliverable we have described the continued implementation of a variety of different approaches to the process of ontology evolution that were described in previous NeOn deliverables. These approaches make use of different kinds of information as their source data: annotations, unstructured text, folksonomies, ontologies, databases and queries. These approaches are all implemented as plugins to the NeOn toolkit. In the case of SPRAT and SARDINE, the GATE webservices plugin which encompasses them is loosely coupled with the toolkit. This means that these applications require a method for ontology change management, in order to synchronise changes made to the ontology in the NeOn toolkit with those made in GATE where the ontology is updated by these applications. We have finalised the ontology change management tool in GATE and have tested its interoperability with the toolkit in order to ensure full compatibility. As a result, changes made to the ontology in GATE can be merged with an existing ontology in the toolkit and vice versa, which enables distributed ontology development. The change management process has been fully integrated with the GATE webservice applications, which means that it is automatically applied when the applications are run, and a change log is automatically saved at the end of the process.

It is possible to make use of the data-driven and socially-driven applications (SPRAT, SARDINE, Evolva and FLOR) as input to the query-driven application RELExO. In particular, we envision a meta knowledge expert which facilitates the integration and evaluation of the axioms generated by any of these applications. This expert could selectively import from the output of these applications all the potentially useful extensions of the overall ontology that is being constructed thereby taking into account the provenance (e.g. uncertainty) information associated with the individual axioms. The implementation and evaluation of such a meta knowledge expert will form part of future work, as will continued development of all the tools, in order to improve their results further.

Bibliography

[AHO06] H. Alani, S. Harris, and B. O'Neil. Winnowing ontologies based on application use. In *Proceedings of 3rd European Semantic Web Conference (ESWC'06), Montenegro*, 2006.

[ATBC05] N. Aswani, V. Tablan, K. Bontcheva, and H. Cunningham. Indexing and Querying Linguistic Metadata and Document Content. In *Proceedings of Fifth International Conference on Recent Advances in Natural Language Processing (RANLP2005)*, Borovets, Bulgaria, 2005.

[BHSV06] Stephan Bloehdorn, Peter Haase, York Sure, and Johanna Voelker. Ontology evolution. In *Semantic Web Technologies - Trends and Research in Ontology-based Systems*, pages 51–70. John Wiley & Sons, June 2006.

[CMT00] H. Cunningham, D. Maynard, and V. Tablan. JAPE: a Java Annotation Patterns Engine (Second Edition). Research Memorandum CS–00–10, Department of Computer Science, University of Sheffield, November 2000.

[CRF03] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03)*, 2003.

[dBG$^+$07] M. d'Aquin, C. Baldassarre, L. Gridinoc, M. Sabou, S. Angeletou, and E. Motta. Watson: Supporting next generation semantic web applications. In *Proceedings of WWW/Internet conference*, 2007.

[dCGPPSF08] G. Aguade de Cea, A. Gómez-Pérez, E. Montiel Ponsoda, and M-C. Suárez-Figueroa. Natural language-based approach for helping in the reuse of ontology design patterns. In *Proceedings of the 16th International Conference on Knowledge Engineering and Knowledge Management Knowledge Patterns (EKAW 2008)*, Acitrezza, Italy, September 2008.

[dSSS09] M. d'Aquin, A. Schlicht, H. Stuckenschmidt, and M. Sabou. Criteria and evaluation for ontology modularization techniques. In *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, pages 67–89. Springer-Verlag, 2009.

[ES07] J. Euzenat and P. Shvaiko. *Ontology Matching*. 2007.

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

[GKT07] Todd J. Green, Gregory Karvounarakis, and Val Tannen. Provenance Semirings. In *PODS*, pages 31–40, 2007.

[Har09] Harry Halpin. Provenance: The Missing Component of the Semantic Web. Heraklion, Greece, 2009. CEUR Workshop Proceedings, online CEUR-WS.org/Vol-447/paper1.pdf.

[Hea92] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Conference on Computational Linguistics (COLING'92)*, Nantes, France, 1992. Association for Computational Linguistics.

[HS05] P. Haase and L. Stojanovic. Consistent evolution of OWL ontologies. In *Proceedings of the Second European Semantic Web Conference, Heraklion, Greece, 2005*, pages 182–197, 2005.

[JQH08] Q. Ji, G. Qi, , and P. Haase. A relevance-based algorithm for finding justifications of DL entailments. Technical report, University of Karlsruhe, 2008.

[Kle04] M. Klein. *Change Management for Distributed Ontologies*. PhD thesis, Vrije Universiteit in Amsterdam, 2004.

[MAP$^+$09] D. Maynard, N. Aswani, W. Peters, F. Zablith, and M. D'Aquin. Advanced methods for change propagation between networked ontologies and metadata. Technical Report D1.5.3, NeOn Project Deliverable, 2009.

[MFP09a] D. Maynard, A. Funk, and W. Peters. NLP-based support for ontology lifecycle development. In *CK 2009 – ISWC Workshop on Workshop on Collaborative Construction, Management and Linking of Structured Knowledge*, Washington, USA, October 2009.

[MFP09b] D. Maynard, A. Funk, and W. Peters. SPRAT: a tool for automatic semantic pattern-based ontology population. In *International Conference for Digital Libraries and the Semantic Web*, Trento, Italy, September 2009.

[MFP09c] D. Maynard, A. Funk, and W. Peters. Using Lexico-Syntactic Ontology Design Patterns for ontology creation and population. In *WOP 2009 – ISWC Workshop on Ontology Patterns*, Washington, USA, October 2009.

[MS00] Alexander Maedche and Steffen Staab. Discovering conceptual relations from text. In Werner Horn, editor, *ECAI 2000, Proceedings of the 14th European Conference on Artificial Intelligence (ECAI)*, pages 321–325. IOS Press, 2000.

[NCLM06] N. F. Noy, Abhita Chugh, William Liu, and Mark Musen. A framework for ontology evolution in collaborative environments. In *The Semantic Web - ISWC 2006*, pages 544–558. 2006.

[NLH07] V. Novacek, L. Laera, and S. Handschuh. Semi-automatic integration of learned ontologies into a collaborative framework. In *International Workshop on Ontology Dynamics (IWOD-07)*, 2007.

[NM04] N. F. Noy and M. A. Musen. Specifying ontology views by traversal. In *The Semantic Web âĂŞ ISWC 2004*, pages 713–725. 2004.

[OGG07] K. Ottens, M. Gleizes, and P. Glize. A multi-agent system for building dynamic ontologies. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–7, Honolulu, Hawaii, 2007. ACM.

[PGS$^+$08] V. Presutti, A. Gangemi, D. Stefano, G. Aguado, M.C. Suárez-Figueroa, E. Montiel-Ponsoda, and M. Poveda. A library of Ontology Design Patterns: reusable solutions for collaborative design of networked ontologies. Technical report, NeOn project deliverable D2.5.1, 2008.

[PHYd08] R. Palma, P. Haase, W. Yimin, and M. d'Aquin. Propagation models and strategies. Technical report, NeOn Project Deliverable, 2008.

[QS09] Guilin Qi and Simon Schenk. NeOn Deliverable D3.1.4. Technical report, The NeOn Project, 2009.

[Sch05] Karin Kipper Schuler. *VerbNet: A broad-coverage, comprehensive verb lexicon*. PhD thesis, University of Pennsylvania, 2005.

[SdCd+09] M. Sabou, G. Aguado de Cea, M. d'Aquin, E. Daga, H. Lewen, E. Montiel-Ponsoda, V. Presutti, and M. C. Suarez-Figueroa. D2.2.3 methods and tools for the evaluation and selection of knowledge components. Technical report, NeOn Project Deliverable, 2009.

[SdM08] M. Sabou, M. d'Aquin, and E. Motta. Exploring the semantic web as background knowledge for ontology matching. *Journal on Data Semantics*, (XI), 2008.

[Ser08] Baris Sertkaya. Explaining user errors in knowledge base completion. In Franz Baader, Carsten Lutz, and Boris Motik, editors, *Description Logics*, volume 353 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[Sto04] L. Stojanovic. *Methods and Tools for Ontology Evolution*. PhD thesis, FZI - Research Center for Information Technologies at the University of Karslruhe, 2004.

[VB08] Johanna Völker and Eva Blomqvist. D3.8.1 prototype for learning networked ontologies. Technical Report D3.8.1, Institute AIFB, University of Karlsruhe, FEB 2008. NeOn Deliverable.

[VB09] Johanna Völker and Eva Blomqvist. D3.8.2 evaluation of methods for contextualized learning of networked ontologies. Technical Report D3.8.2, Institute AIFB, University of Karlsruhe, FEB 2009. NeOn Deliverable.

[VPST05] D. Vrandecic, H. S. Pinto, Y. Sure, and C. Tempich. The DILIGENT knowledge processes. *Journal of Knowledge Management*, 9(5):85–96, 2005.

[VR08a] Johanna Völker and Sebastian Rudolph. Fostering web intelligence by semi-automatic OWL ontology refinement. In *Proceedings of the 7th International Conference on Web Intelligence (WI)*, pages 454–460. IEEE, December 2008. regular paper.

[VR08b] Johanna Völker and Sebastian Rudolph. Lexico-logical acquisition of OWL DL axioms – An integrated approach to ontology refinement. In *Proceedings of the 6th International Conference on Formal Concept Analysis (ICFCA)*, volume 4933 of *Lecture Notes in Artificial Intelligence*, pages 62–77. Springer, February 2008.

[VTSFGP+08] B. Villazón-Terrazas, M. Suárez-Figueroa, A. Gómez-Pérez, A. García-Silva, and D. Maynard. Methods and tools for re-engineering non-ontological resources. Technical Report D2.2.2, NeOn Project Deliverable, 2008.