NeOn: Lifecycle Support for Networked Ontologies

Integrated Project (IST-2005-027595)

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

D2.2.3 Methods and Tools for the Evaluation and Selection of Knowledge Components

Deliverable Co-ordinator: **Marta Sabou**

Deliverable Co-ordinating Institution: **Open University (OU)**

Other Authors: **Guadalupe Aguado de Cea (UPM), Mathieu d’Aquin (OU), Enrico Daga (CNR), Holger Lewen (UKARL), Elena Montiel-Ponsoda (UPM), Valentina Presutti (CNR), Mari Carmen Suárez-Figueroa (UPM)**

This deliverable describes a set of methods and tools that support evaluation and selection of various knowledge components (ontologies, ontology modules, ontology design patterns, ontology statements) within the context of the NeOn project.

Document Identifier:	NEON/2007/D2.2.3/v1.0	Date due:	February 28, 2009
Class Deliverable:	NEON EU-IST-2005-027595	Submission date:	February 28, 2009
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 Kolenz-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:

- Open University (OU)
- Universidad Politécnica de Madrid (UPM)
- Universität Kolenz-Landau (UKO-LD)
- Consiglio Nazionale delle Ricerche (CNR)
- Universität Karlsruhe – TH (UKARL)
- University of Sheffield (USFD)
- Institut 'Jožef Stefan' (JSI)
- Food and Agriculture Organization of the United Nations (FAO)

Change Log

Version	Date	Amended by	Changes
0.1	08-01-2009	Marta Sabou	First draft of the deliverable structure.
0.2	26-02-2009	Marta Sabou	First complete draft.
0.3	11-03-2009	Marta Sabou	Final version to be sent for QA.
0.4	08-04-2009	Marta Sabou	Final version taking into account QA comments.

Executive Summary

This deliverable presents a set of methods for the evaluation and selection of a variety of knowledge components. Indeed, unlike the previous deliverable in this task, D2.2.1 [SAd$^+$07], one of the goals of this deliverable is to extend the issues of evaluation and selection to other knowledge components besides ontologies. We present methods and techniques at various levels of maturity. The first two parts of the deliverables describe methods which have matured to a level where they have been included in various tools associated with the NeOn project. The final part presents novel research on evaluating ontology statements.

An interesting finding is that different project partners have pursued different approaches under the broad umbrella of evaluation of knowledge components. These approaches are mainly influenced by the particular expertise of the partners that developed them. We distinguish three main approaches, which are also reflected in the structure of this deliverable by corresponding to its three main parts.

Firstly, CNR's work on ontology design patterns is reflected in their XD design pattern based evaluation framework (Chapter 2). This has been successfully applied to the Fisheries use case, where the authors were able to detect that the investigated ontology did not conform to a given competency question. Additionally, a side-effect of CNR's work on the ontologydesignpatterns.org portal is a wiki-based methodology (Evaluation Wiki Flow) for a review-based evaluation of any type of knowledge components (Chapter 3). This methodology currently supports the portal and a user evaluation is planned.

Second, UKARL' s work on open rating systems focuses on evaluating ontology components based on user reviews and the trust level that other users express in these reviews. The performance of this system is formally evaluated from different perspectives and conclusions are drawn about its optimal setup. We also present the adaptation of this system in the context of the NeOn Watson plugin, resulting in an initial system that still needs evaluation (Chapter 5).

Finally, OU and UPM have joined efforts in the task of evaluating the correctness of ontology statements. UPM investigated the use of natural language based techniques, in particular that of various types of lexical syntactic patterns for predicting the correctness of a given statement. The general conclusion was that, while this technique provides promising results for this pilot study, larger scale experiments need to be performed to reach a final conclusion. The OU has proposed a solution which relies on online available ontologies and in the way they describe knowledge to judge correctness. This solution has been experimentally explored, showing that methods which explore ontologies as knowledge components work better than those which regard their collection as a corpus. Given the initial promising results, a formal framework for measuring agreement in online ontologies was proposed.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Work package 2, Collaborative Aspects for Networked Ontologies, investigates collaborative aspects of ontology development and reuse. The main objective of WP2 is to analyze and describe (by means of a suitable meta-model) the activities underlying collaborative design of networked ontologies, and to produce appropriate methods and tools to support the related workflow, by focusing on the collaborative aspects underlying the work of a knowledge community.

Task T2.2 within WP2 provides methods and tools to support re-engineering, evaluation and selection in the context of building networked ontologies. A particular kind of collaborative ontology development is that of reusing existing knowledge sources. Reuse can happen at different levels of granularity - entire ontologies, ontology modules or even individual ontology axioms. One of the novel, defining characteristics of the NeOn toolkit is that it supports axiom-level reuse from online ontologies through the Watson plugin. The process of building an ontology by reusing axioms from several online ontologies leads to interconnecting these ontologies into an ontology network.

Task T2.2. has two subtasks, one focusing on re-engineering and another focusing on evaluation and selection. This deliverable provides an overview of methods and tools for the evaluation and selection of knowledge components. The work presented here is a continuation of work described in D2.2.1 [SAd$^+$07], but differs in two main aspects:

Focus on knowledge components rather than entire ontologies. One of the conclusions of D2.2.1 was that evaluation and selection should be performed not only at ontology level but also at the level of more fine-grained knowledge components such as ontology modules, ontology design patterns or simply ontology statements. Indeed, the frameworks described in Part I and Part II are generic and can be used to evaluate a wide range of knowledge component types. Further, Part III, focuses entirely on methods for evaluating ontology statements.

Focus on methods that are concrete components of the NeOn tool suite. Another goal of this deliverable was to integrate work described in D2.2.1 in concrete tools developed within the NeOn project. Indeed, Part I and Part II present works that have matured to a level where they became parts of NeOn tools.

This deliverable provides an overview of the efforts that different project partners have pursued under the broad umbrella of evaluation of knowledge components. As a result, we report on a variety of different approaches which were influenced by the particular expertise of the partners that developed them. In particular, we distinguish three main approaches, which are also reflected in the structure of this deliverable by corresponding to its three main parts. Firstly, CNR's work on ontology design patterns is reflected in their XD design pattern based evaluation framework. Additionally, a side-effect of their work on the ontologydesignpatterns.org portal is a wiki-based methodology for a review-based evaluation of any type of knowledge components. Second, UKARL' s work on open rating systems focuses on evaluating ontology components based on user reviews and the trust level that other users express in these reviews. The performance of this system is formally evaluated and we also present its adaptation in the context of the NeOn Watson plugin. Finally, OU and UPM have joined efforts in the task of evaluating the correctness of ontology statements. UPM investigated the use of natural language based techniques, in particular that of various types of lexical syntactic patterns for predicting the correctness of a given statement. The OU has proposed a solution which relies on online available ontologies and in the way they describe knowledge to judge correctness. This solution has been experimentally explored and then formalized in a framework.

1.1 Overview

Following this introductory chapter, the material in this deliverable is structured in three main parts each describing methods that subscribe to a certain paradigm or that focus on the evaluation of a certain type of knowledge component. The mature techniques that have been applied in the context of concrete tools are presented first, followed by more recent work.

Part I: Pattern Based Evaluation of Ontology Components describes methods and tools inspired by work on ontology design patterns.

Chapter 2 presents Evaluation XD (EXD), a method for ontology evaluation inspired by the paradigm of eXtreme Design (XD) which explores ontology design patterns. This method explores the *oQual* ontology evaluation framework which has been previously described in D2.2.1.

Chapter 3 details Evaluation WikiFlow, a wiki-based tool that supports workflows for evaluating any kind of knowledge components e.g., ontologies, vocabularies, articles, ontology design patterns, etc. While generic to any knowledge component, this chapter describes an instantiation of WikiFlow for ontology design patterns in the context of the ontologydesignpatterns.org initiative.

Part II: Trust-based Evaluation of Ontology Components illustrates two different methods that rely on the notion of trust in order to evaluate ontology components.

Chapter 4 describes the implementation of a topic-specific open rating system (TS-ORS, reported in D2.2.1.) in the context of the Watson ontology search engine. The resulting system allows users to rate various parts of an ontology (the entire ontology, a given property of the ontology) and to express their trust in other reviews. Based on this trust information, a web of trust is computed that allows the ranking of reviews for each ontology - property combination and the derivation of a final rating for an ontology.

Chapter 5 exploits trust relations implicitly established when reusing an ontology statement rather than trust relations explicitly declared by agents. These relations are collected from a community of users and propagated on the conceptual structures of ontologies in order to derive an overall ranking both for individual ontology statements and the ontology as a whole.

Part III: Methods for Evaluating Ontology Statements describes novel work on evaluating ontology statements.

Chapter 6 contains some introductory considerations to this topic, such as the overall motivation and the research methodology that has been followed.

Chapter 7 presents investigations into using Natural Language based techniques for validating the correctness value of an ontology statement. To that end, this chapter reuses many of the Lexical Syntactic Patterns associated with ontology design patterns.

Chapter 8 closes this part with a collection of methods that use the redundancy of information available on the Web and the Semantic Web to evaluate the correctness of an ontology statement. We provide some experimental investigations of these methods and then derive a formal framework which extends to the topic of measuring agreement and disagreement in online ontologies.

1.2 Integration with the Rest of the Project

The work reported in this deliverable has been tightly integrated with the efforts of other work packages in the project. In particular:

WP1 The work reported in Chapter 4 on the integration of TS-ORS with Watson forms the bases of the Cupboard system developed within WP1.

WP2 This work has been strongly influenced by work on ontology design patterns performed in task T2.5. Chapter 2 describes an ontology evaluation method based on ontology design patterns. Chapter 3 presents a generic evaluation framework using wikis that has been tested in the context of a repository of ontology design patterns. Finally, Chapter 7 describes a set of tools for validating ontology statements based on lexical syntactic patterns associated to ontology design patterns.

WP3 The work presented in Part III on the evaluation of ontology statements is of direct relevance to ontology matching techniques that form part of the Alignment server in WP3.

WP7 Chapter 2 gives an account of how the presented evaluation framework has been applied to a concrete case in the context of WP7.

Part I

Pattern Based Evaluation of Ontology Components

Chapter 2

Pattern-based ontology evaluation (CNR)

This chapter describes *Evaluation XD (EXD)*, a method for evaluating ontologies based on ontology design good practices i.e. ontology design patterns (ODPs) according to [PGD+08].
ODPs are used as components for evaluating and validating ontologies with respect to different dimensions. Although the method is not meant to be comprehensive, it can be used in combination with other existing approaches for evaluating and improving ontologies.
The chapter is organized as follows: in the next section we briefly introduce *eXtreme Design (XD)*, a family of methods that provide the context for EXD. Section 2.2 gives a brief introduction to *oQual* [GCCL06], a formal framework for ontology validation, on which EXD relies. EXD and its components are then described in Section 2.3 and an example of its application of its application in the *Fishery* domain is discussed in Section 2.4.

2.1 eXtreme Design (XD): a brief introduction

EXD is part of a more general approach to ontology design that has been developed in NeOn WP2, namely *eXtreme Design (XD)*, and that will be described in depth in forthcoming Deliverable D2.5.2 (due at M42). With the name XD we identify a family of methods based on the application, exploitation, and definition of ontology design patterns (ODPs) [PGD+08], for solving problems related to ontology design. Consider for example an ontology project that is developed by following the NeOn methodology (cf. Deliverable 5.3.2). The XD approach can be used in order to work out some/all the activities of the NeOn methodology. In other words, ODPs represent the ontology project's *solution space*, which is used as the main knowledge source for addressing ontology design issues e.g. reengineering, evaluation, construction.
From the project's *solution space*, a number of ODPs are selected by matching the actual (local) problems, namely the ontology project's *problem space*, typically expressed in terms of competency questions, with General Use Cases (GUC) that the ODPs are supposed to be a solution for [Gan05]. The general approach is schematized in Figure 2.1. The selected ODPs are then used accordingly to specific guidelines as explained in WP5 Deliverable 5.4.2 and in greater detail in the forthcoming D2.5.2.

2.2 oQual: a brief introduction

Evaluation XD (EXD) is a practical method for evaluating ontologies based on ontology design patterns. It is inspired and can be considered a specific application of a general formal framework for evaluating and validating ontologies named *oQual* [GCCL06], briefly introduced here. The intuition behind EXD is that ODPs provide "gold standards" to which ontologies and/or part of them can be compared to in order to decide if they comply to specific needs.
The oQual framework does not describe a specific evaluation method, but identifies dimensions and com-

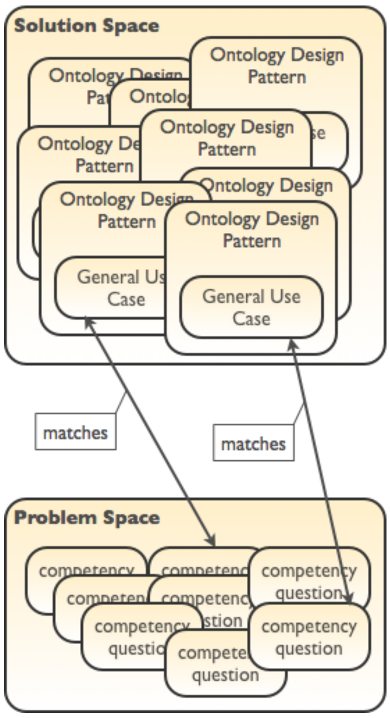


Figure 2.1: The eXtreme Design general approach. The project is characterized by two sets: the *problem space*, which is composed of the actual problems that have to be addressed during the project, and the *solution space*, which is made up of successful reusable solutions. Each solution is an Ontology Design Pattern and is associated to a General Use Case.

ponents that constitute and have to be taken into account for ontology evaluation and validation. oQual is characterized as follows:

- O_2: a model that formally characterizes ontologies as *semiotic objects* that are information of a special kind and feature (i) a graph-like structure, and (ii) an intended conceptualization.
- oQual [GCCL06], that models ontology validation as a diagnostic task, and provides the means to devise the best set of criteria for choosing an ontology over others in the context of a given project.
- Three dimensions for evaluating ontologies: Structural, Functional, Usability-Profiling.
- *Black box* and *glass box* measurement methods for evaluating precision, recall, and accuracy of an ontology:
  - Black box methods require rational agents, because they don't explicitly use knowledge of the internal structure of an expertise (see [Ald05] for more details).
  - Glass box methods require a data set that "sample" that knowledge on which basis we can treat the internal structure of those data as if it were the internal structure of an expertise.

2.3 Evaluation XD (EXD)

According to the oQual framework, EXD provides a practical approach for evaluating ontologies relatively to three main dimensions: structural, functional, and usability-profiling. The intuition behind EXD is that ODPs are used as "contextualized gold standards", to which ontologies and/or part of them can be compared to in order to decide if they comply to specific needs. Figure 2.2 depicts the classification of ontology design patterns according to [PGD$^+$08].

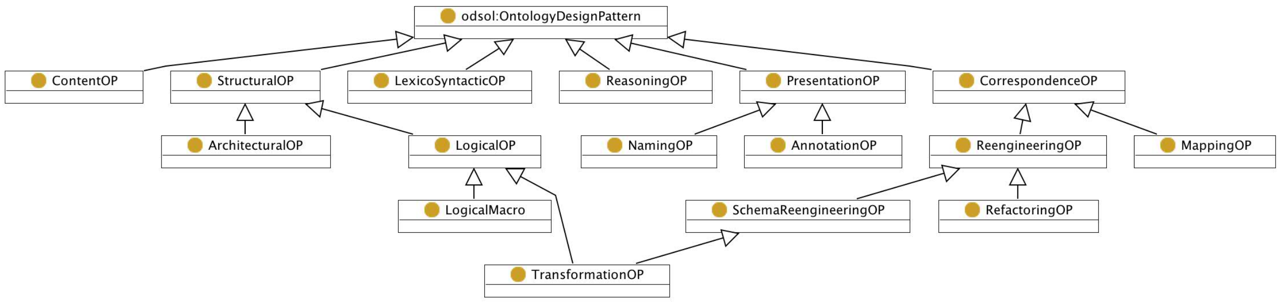


Figure 2.2: Types of Ontology Design Patterns.

Depending on the requirements that the ontology is expected to address, the appropriate ODPs are selected and included in the set of gold standards used for the specific case.

2.3.1 Evaluation for selection *versus* evaluation for improvement

Based on practical experience within NeOn case studies and other realistic projects, we can distinguish between two main situations characterizing the application of EXD for evaluating ontologies: evaluation for selection, and evaluation for improvement.

- Evaluation for selection: refers to the situation where existing ontologies are evaluated with respect to the three dimensions, in order to select the most appropriate one.

- Evaluation for improvement: refers to the situation where a certain version of an ontology is evaluated against good practices i.e. ODPs, with the aim of performing revisions leading to new improved versions of the ontology. This is usually an iterative process, and has been the case of the NeOn case study of ontologies for the fishery domain (see also deliverable 7.2.3 due at M36).

Although the method and criteria applied are the same, the main difference in such two cases is the goal of the evaluation: in the first case, an ontology is either selected or discarded; in the second case, the evaluation produces a set of review comments to be addressed in a revised version of the ontology.

2.3.2 Global *versus* local evaluation

Before describing EXD application with respect to the three dimensions: structural, functional, and usability-profiling, it is important to remark the difference between *global* and *local* evaluation of an ontology.
An ontology is always meant to encode some conceptualization of a certain domain of knowledge, with respect to specific tasks. This means that, depending on such domain and tasks, the same ontology can be found more or less adequate, although having the same structural and usability-profiling measures.
In other words, on one hand there are *global* measures that can be computed regardless of the specific domain and tasks, on the other hand there are *local* measures that are dependent on the specific domain and tasks that the ontology has to cover.
For example, if we want to evaluate the adequacy of an ontology to be used in an application for the management of *invoices*, we would not look at ontologies encoding knowledge about *conference organization*, even if the latter has high structural and usability-profiling quality. This leads to the fact that the *functional* aspect, reflecting the *local* criteria, is the most important dimension, the first in a possible priority scale of evaluation dimensions.
Figure 2.3 sketches the main steps that characterize EXD.

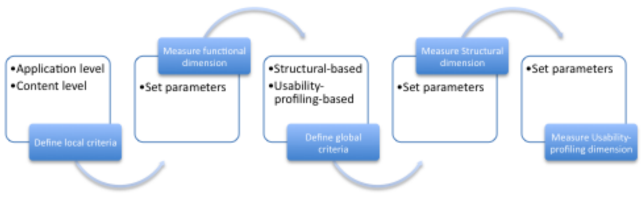


Figure 2.3: EXD main steps. The first step is the definition of local criteria for evaluation, they are specified based on two levels: application and content. Based on local criteria, in the second step, the functional dimension is measured according to parameters that are set according to the specific evaluation context. The third step requires the definition of the global evaluation criteria related to structural and usability-profiling dimensions. Based on global evaluation criteria, the fourth and fifth steps deal with measurements of the structural and usability-profiling dimensions, respectively, both according to contextualized setting of measure parameters.

2.3.3 Measuring the functional dimension

The functional dimension of an ontology is related to the main purpose of an ontology, the conceptualization that the ontology has to specify. Such specifications are always approximate: on one hand they depend on a rational agent that conceives that conceptualization, on the other hand, they depend on the semantic space that formally encodes it.
This aspect has been highlighted in [GCCL06] and is approached there by assuming that precision, recall, and accuracy of an ontology can be measured against: (i) experts' judgment, or (ii) a data set assumed as a qualified expression of experts' judgement, and by applying two possible measurement methods, namely black box and glass box (see section 2.2). (ii) is perfectly inline with EXD approach. Specifically, EXD provides means to measure the ontology quality against two aspects: *Task*, and *Modularity*, both based on glass box methods.

Task

What has to be supported by the ontology? This aspect deals with measuring an ontology according to its fitness to some goals, preconditions, postconditions, constraints. According to [PGD$^+$08], EXD proposes to evaluate the ontology against so called *Content ODP*, *Correspondence ODPs*, and *External Architectural ODPs* based on the assumption that: *the task of an ontology is expressed by means of a set of competency questions* [GF94].
Competency questions can be elicited at different levels, here we consider two of them: application level and content level.

- *Application level*: the domain of competency questions is the application context where the ontology has to be deployed. Typically, such competency questions are associated with external architectural solutions and evaluation is performed against External Architectural ODPs.
- *Content level*: the domain of competency questions is the domain whose conceptualization is expected to be encoded by the ontology. Such competency questions are associated with domain-related design choices, and evaluation is performed against Content ODPs.

Application level For the application level, consider the following case:

- The ontology is used in order to define at runtime the configuration of the user environment of a software application. The application covers different possible domain scenario e.g. workflow definition, user profile specification, document template definition etc. The application should be able to load only

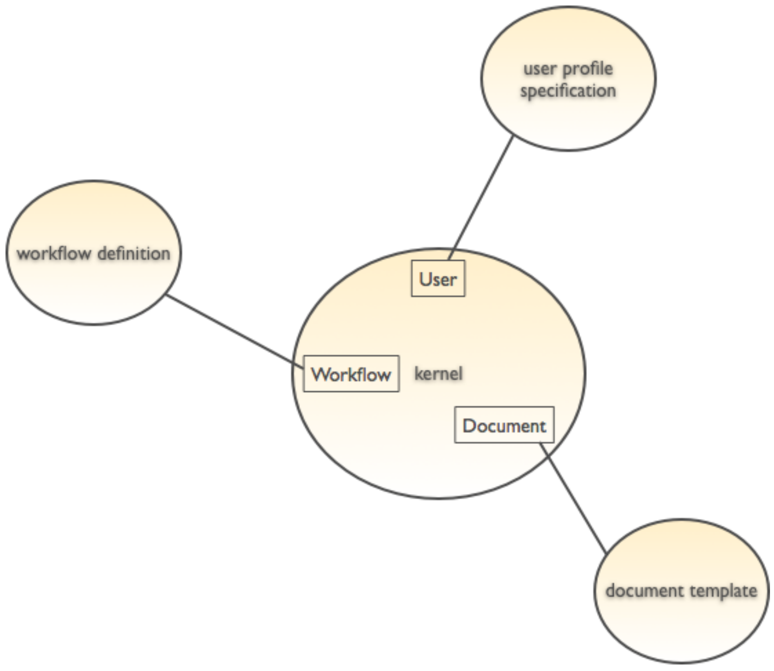


Figure 2.4: An ontology designed by means of the *corolla* Architectural ODP. The kernel module defines core concepts, such as *Workflow*, *User*, *Document*, etc., and their mutual dependences, while each petal module defines specific properties and additional classes useful for e.g. workflow definition, user profile specification, document template definition, etc. Each petal module imports the kernel module and further locally axiomatizes at least one concept defined in the kernel module.

.

the specific part of the ontology that is needed to the current purpose, hence the ontology external architecture should be modular.

A possible External Architectural pattern that can be used as gold standard is the *corolla* architecture. Such Architectural ODP is composed of a *kernel* module, which includes the definition of core concepts that are common to all other modules, and a set of *petal* modules, each defining a specific domain area of interest. An example of how the external architecture of a suitable ontology for the considered case is shown in Figure 2.4. The *kernel* module defines core concepts and is imported by all *petal* modules. Petal modules refine the axiomatization of at least one of the core concept.

Content level The content level represents the key aspect of the functional dimension, but there are no exhaustive measures to be applied for its evaluation. This is due to content being depend on the specific domain, and for a same domain there can be several different tasks, in turn influencing the measurement reference parameters. Nevertheless, there are many studies and measures that can be used in a combined way in order to work out this task. In [BGM05], a reasonable classification of approaches to ontology evaluation is made which identifies four typical approaches that can be related to some extent to the content level of functional dimension:

1. the comparison of the ontology to a "gold standard", as for example in [MS02], where a set of similarity measures are defined;
2. the evaluation of results of usage of the ontology in a certain application as for example in [PM04], where ontologies are evaluated against specific tasks mainly to the aim of monitoring their improvements or degradation during their evolution;

3. the comparison of the ontology with some source of data, typically made by exploiting NLP-based techniques for measuring coverage;

4. user-based evaluation as in [LTGP04], where ontology selection is based on verifying a set of standard-like criteria relying on expert's judgment.

EXD does not propose a new specific measure, while it provides ontology designers with a pragmatic approach that can be associated to some extent with all of the above mentioned approaches, meaning that it allows, according to [GCCL06], to set custom functional requirements and to apply the corresponding adequate measures, based on the local needs.
With respect to 1., EXD relies on the use of ODPs as constituting a gold standard. The gold standard elements are selected by the ontology designer based on the specific domain and tasks to be addressed, and used for comparison with existing ontologies. With respect to 2., EXD relies on the use of unit tests as representatives of the task to be addressed by the ontology. With respect to 3., and 4. no specific assumptions are made.
We represent an ontology as a set of two elements:

$$O = \{V, A\} \tag{2.1}$$

where V is the set of terms composing its vocabulary, and A is the set of axioms it encodes. In the rest of the section, by $O.V$, we refer to the vocabulary of the ontology O, and by $O.A$ to the set of axioms encoded by the ontology O. We also represent the set of competency questions elicited by an expert, which has to be covered by an ontology with:

$$Q_l = \{q_{l1}, ..., q_{ln}\} \tag{2.2}$$

Note that the l subscript stays for *local*, meaning that Q_l includes the competency questions representing the local problem. With regard to Content ODPs, they are associated with competency questions that represent their General Use Case (GUC). Each Content ODP's set of competency question can be represented as a set

$$Q_{name} = \{q_{name-1}, ..., q_{name-n}\} \tag{2.3}$$

where the subscript $name$ indicates a Content ODP name.
Finally, we represent with *gold_standard* the set of selected Content ODPs selected for comparison in the specific evaluation context. Such set is built by matching Q_l with the sets of type Q_{name} associated with the available Content ODPs. Each Content ODP is accompanied by a Q_{name} set representing its General Use Case.

Evaluation of an existing ontology with no explicit usage of ODPs In this case, we assume that there are no explicit statements in the ontology that allow the identification of Content ODPs in its formal encoding. It is therefore necessary to refer to three resources: (i) the specific portion of domain vocabulary needed in the specific evaluation context, named *desired_vocabulary*, (ii) the key axioms that should be encoded by the ontology, named *key_axioms*, and (iii) a set of unit test queries based on Q_l, named *unit_tests*.
Based on such resources three functional criteria are considered as measures for the evaluation of an ontology O:

- the percentage of vocabulary coverage: a measure indicating the extent to which $O.V$ covers *desired_vocabulary*;

- the percentage of key axioms matching: a measure indicating the number of axioms in $O.A$ that matches axioms in *key_axioms*;

- the percentage of successful runs of *unit_tests* against O.

In the case of evaluation for improvement purposes, an additional resource is developed, the *gold_standard* set. The Content ODPs in *gold_standard* can be used in order to improve a next version of the ontology.
It is worth to remark that the application of the second approach can be difficult. For example, it can be the case that there is not yet a set of Content ODPs suitable for solving the local problem, or that the ontology to be evaluated is huge and it is not trivial to identify the part of the ontology that specifies the conceptualization corresponding to the selected competency questions. On the other hand, the ontology may implicitly include Content ODPs (they are easily identifiable).
It is important to notice that candidate Content ODPs for the definition of the *gold_standard* can be associated to a Q_{name} related to a domain that is more general than the domain of the ontology O under evaluation. For example, if O is meant to address the domain of scientific conferences and their organization, candidate Content ODPs can be general patterns such as: *participation*, *nary-participation*, *situation*, *object-role*, etc., which respectively specify: the conceptualization of an entity participating to an event; an entity participating to some event, in a certain place, at a certain time, involving other objects; a situation which is the setting of different entities; an object playing a certain role. It might be noticed that such Content ODPs are not specific for the domain of scientific conferences and their organization, but express typical scenarios that occur in this domain. Section 2.4 shows an example in the NeOn case study of Fishery domain that will clarify a situation like this, that often occurs in real scenarios.

Evaluation of an ontology built by using XD method If the ontology O to be evaluated is known to be built according to XD method, hence e.g. by explicitly using Content ODPs, the ontology designer proceeds by selecting a set of Content ODPs that constitute the *gold_standard*. The evaluation is then performed by matching the Content ODPs included in the *gold_standard* with the ones used in the design of O. The ontology O is then assigned with a measure reflecting such matching, it indicates the number of Content ODPs used in O that are also included in the *gold_standard*.

2.3.4 Measuring the structural dimension

The structural dimension of an ontology can be evaluated against several measures as it is exemplified in [GCCL06] e.g. topological properties such as *depth*, *breath*, etc.
With reference to the classification depicted in Figure 2.2 defied in [PGD$^+$08], EXD proposes to compare the structure of an ontology by comparing it with so called *Structural ODPs* and *Reasoning ODPs*. In particular, for Strucural ODPs, *Logical ODPs*, and *internal architectural patterns*[1] are suitable to this aim.

Comparison with logical macros A Logical ODP is a formal expression, whose only parts are expressions from the logical vocabulary of e.g. OWL DL, that solve a problem of expressivity. For example, the **universal+existential** OWL macro, described in [Vra05], allows to model a recurrent intuitive quantification, as from the following example: "all horses have parents that are horses", requiring both an `owl:allValuesFrom` restriction (only horses can be parents of horses) and a `owl:someValuesFrom` restriction (all horses have parents) [PGD$^+$08]. Such kind of patterns can be used, for example, in order to validate the ontology based on structural quality criteria such as the presence of *qualified dense* areas. Such quality criteria can be partly expressed by means of two sample requirements:

- r_1: most object properties (e.g. more than 50%) have domain and range non-empty and different from `owl:Thing`.
- r_2: most classes (e.g. more than 50%) are sub-class of at least one restriction.

In order to evaluate if r_1 is addressed by an ontology, the taxonomy of object properties is compared against the logical macro that expresses the definition of an `owl:ObjectProperty` P with non-empty

[1]Internal architectural pattern are not shown in the Figure 2.2, nevertheless they are defined and described in [PGD$^+$08].

`rdfs:domain` D and `rdfs:range` R, where D is not `owl:Thing` and R is not `owl:Thing`.
On the other hand, in order to evaluate if r_2 is addressed by the ontology under evaluation, classes definition are compared against the logical macros that expresses the definition an `owl:Class` C as an `rdfs:subClassOf` an `owl:Restriction` on an `owl:ObjectProperty` P with value either `owl:allValuesFrom` or `owl:someValuesFrom`.

Comparison with internal architectural ODPs and reasoning ODPs *Architectural ODPs* affect the overall shape of the ontology: their aim is to constrain 'how the ontology should look like'. The evaluation of an ontology along the structural dimension can be performed by comparing its structure with *internal architectural ODPs*. They are defined either in terms of collections of Logical ODPs that have to be exclusively employed when designing an ontology e.g., an OWL species, or the varieties of description logics; or by asserting that a certain property must hold for the ontology.
For example, consider the following requirements:

- r_1: the ontology expressivity must comply the OWL Lite species.
- r_2: the ontology must be consistent.

Requirement r_1 can be validated by comparing the ontology structure with the OWL Lite species architectural pattern, which is defined by the set of logical expressions allowed by OWL Lite. In this case, the validation can be done by applying a reasoning patterns, typically implemented by most standard DL reasoners, which checks the species of the ontology.
Requirement r_2 can be validated by applying a *logical consistency* reasoning ODP. Such validation is substantiated by running a consistency checking procedure implemented by common DL standard reasoners e.g. Pellet, Fact++.

2.3.5 Measuring the usability-profiling dimension

The usability-profiling dimension of an ontology relates to the communication context of an ontology, its documentation and appearance. An ontology profile is a set of annotations. It is very common to underestimate the importance of this dimension. An ontology, as any other "software component" should be well documented in order to ease reuse both from external users e.g. on the Semantic Web, by other organizations with similar needs, and from internal users e.g. "the original designer of the ontology does not work here anymore and a new one inherits the task of maintaining and evolving it".
In order to evaluate usability-profiling of an ontology, EXD proposes to compare the ontology profile to so called *Presentation ODPs*.
For example, possible (non-comprehensive) quality principles can be validated against the following *Presentation ODPs*:

- Ontology name and ontology entity names are defined based on a clear naming convention.
- The ontology purpose is explicitly expressed by an `rdfs:comment`.
- All ontology entities are annotated with an `rdfs:label` whose value is a meaningful name for the entity and its associated standard code language.
- All ontology entities are annotated with a set of `rdfs:label`s whose values are meaningful names for the entity in different languages, explicitly stated by a standard language code.
- All ontology entities are annotated with an `rdfs:comment` that clearly describes the intensional meaning of the entity and its role within the ontology.
- The ontology is associated with a descriptive document that includes general comments and diagrams exemplifying the use of the ontology in a real scenario. The document is referred to in the `rdfs:comment` annotation describing the ontology.

2.4 Applying EXD for validating the functional dimension of and ontology in the Fishery NeOn case study

In this section we discuss a concrete case of EXD used for the evaluation of an ontology for the NeOn case study in the Fishery domain based on a set of competency questions provided by expert users. The ontology we use here as example is a draft ontology we call *fao-application* which is available online[2].
According to what is described in Section 2.3.3, we evaluate the ontology over its functional dimension by using Content ODPs. We proceed as follows:

- Given a competency question we analyze the ontology entities and search by keyword the presence of candidate classes and properties that matches the concepts expressed by the competency question.
- We look also if the ontology explicitly reuses, by means of import properties, Content ODPs.
- We select candidate Content ODPs than would be suitable as modeling solutions.
- If the ontology explicitly reuses Content ODPs we compare them with the selected candidates.
- In case the ontology does not explicitly reuse Content ODPs, or (some of) the ones chosen are different from the selected candidates, we compare the modeling solution applied to the selected candidate Content ODPs, used here as gold standard.
- Based on the set of competency questions we define corresponding unit tests in terms of SPARQL queries according to class and property names defined in the ontology under evaluation.
- We test the ontology against the unit tests.
- We then comment about the adequacy of the modeling solution adopted.

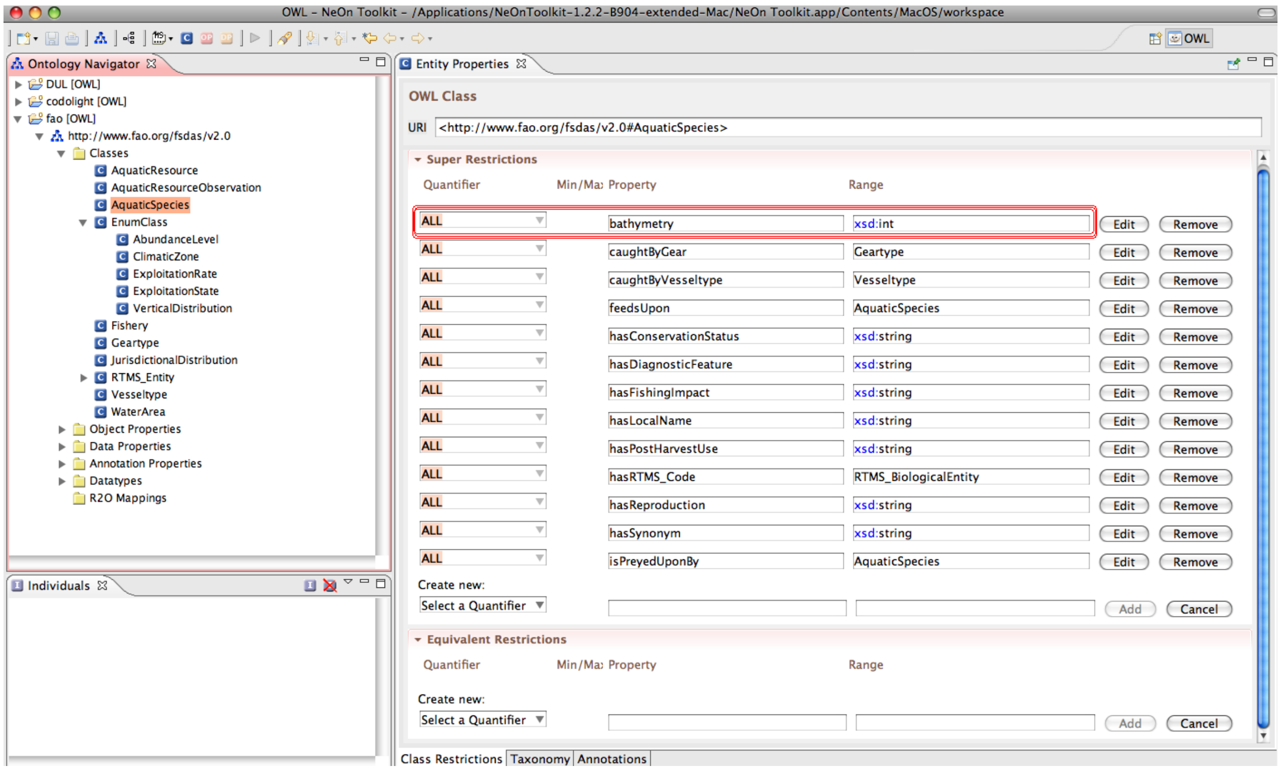
Such procedure would be performed iteratively in presence of a list of competency questions.
For the sake of this example we consider only one competency question among the set provided by the experts. For more details the reader can refer to Deliverable 7.2.3.

Competency Question for the fao-application ontology

Give me the species found below 200 metres for water area "24".

From the fao-application ontology we select the class `AquaticSpecies` and analyze its definition. Figure 2.5 shows such a definition in the NeOn toolkit. From the repository of ontology design patterns available at ontologydesignpatterns.org, we select the Content ODP named *Species Conditions*[3] which intent is “to represent the habitat and bathymetric features that are typical for an aquatic species, in the context of a given water area”. Such Content ODP is modeled as an nary-relation namely `SpeciesConditions`, that relates an `AquaticSpecies` to a `BathymetricRange`, to a `WaterArea`, and to a `Habitat`. Figure 2.6 depicts the definition of the `SpeciesConditions` class in the *Species Conditions* Content ODP. We notice that the fao-application ontology does not explicitly reuse Content ODPs as it does not import any external module.
We compare the modeling solution applied in the fao-application ontology to the *Species Conditions* Content ODP.
It can be noticed that there is a main difference between the two solutions, which deserves some discussion. In the fao-ontology bathymetric range is modeled as an `owl:DatatypeProperty` i.e. `bathymetry`. Such solution could be adequate in the case the value to be expressed was a precise number, or a general string. However, the competency question refers to a range of values i.e. an interval. As such the range is

[2]http://www.ontologydesignpatterns.org/ont/fao/fsdas/f9.owl
[3]http://ontologydesignpatterns.org/wiki/Submissions:SpeciesConditions



OWL Class

URI <http://www.fao.org/fsdas/v2.0#AquaticSpecies>

Super Restrictions

Quantifier	Min/Max	Property	Range		
ALL		bathymetry	xsd:int	Edit	Remove
ALL		caughtByGear	Geartype	Edit	Remove
ALL		caughtByVesseltype	Vesseltype	Edit	Remove
ALL		feedsUpon	AquaticSpecies	Edit	Remove
ALL		hasConservationStatus	xsd:string	Edit	Remove
ALL		hasDiagnosticFeature	xsd:string	Edit	Remove
ALL		hasFishingImpact	xsd:string	Edit	Remove
ALL		hasLocalName	xsd:string	Edit	Remove
ALL		hasPostHarvestUse	xsd:string	Edit	Remove
ALL		hasRTMS_Code	RTMS_BiologicalEntity	Edit	Remove
ALL		hasReproduction	xsd:string	Edit	Remove
ALL		hasSynonym	xsd:string	Edit	Remove
ALL		isPreyedUponBy	AquaticSpecies	Edit	Remove

Create new:
Select a Quantifier — Add — Cancel

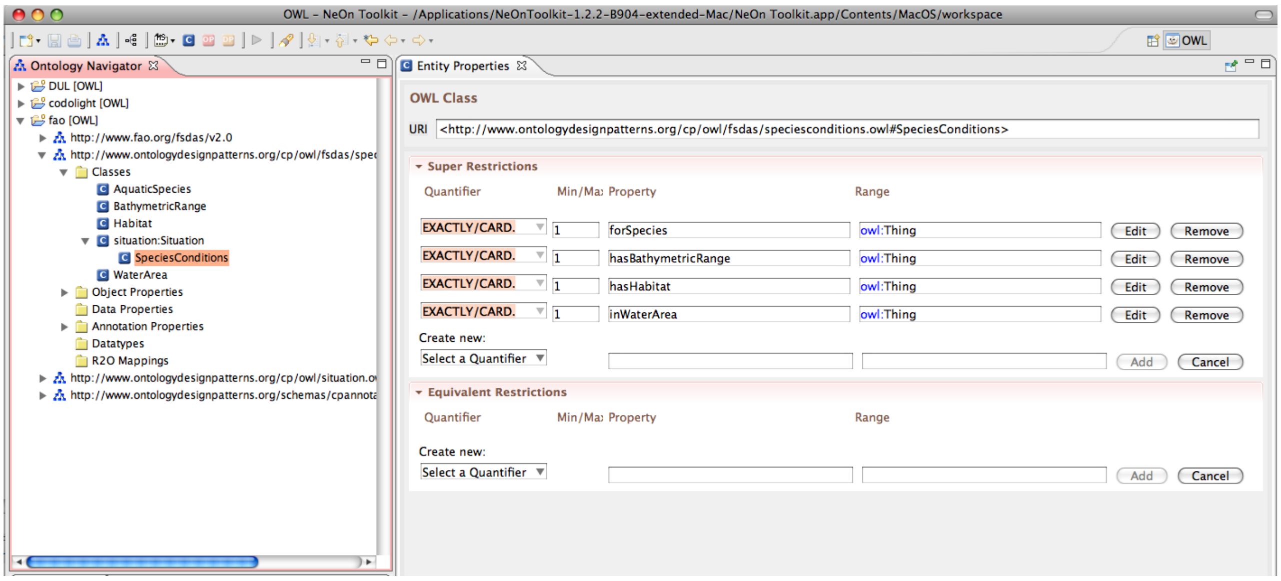
Equivalent Restrictions

Quantifier — Min/Max — Property — Range

Create new:
Select a Quantifier — Add — Cancel

Figure 2.5: The definition of the class `AquaticSpecies` in the fao-application ontology. The picture highlights the datatype property `bathymetry` which is meant to provide “the depth range at which a species is found, generally expressed in meters”.

more likely to be identified by at least two values e.g. the boundaries of the bathymetric interval.
For this reason, the *Species Conditions* Content ODP results to be more suitable as it expresses bathymetric ranges as instances of a class i.e. `BathymetricRange` that can be in the domain of `owl:DatatypeProperty`s identifying the values of the range’s boundaries.
As an additional remark, the competency question requires the identification of a certain species found in a certain bathymetric range in a given water area. The solution adopted by the fao-application ontology requires to define a complex query in order to retrieve such information as, it can be noticed, that there is not a direct relation between the aquatic species and the water area. This information can be retrieved only indirectly by involving the class `AquaticResource`, which is related to `AquaticSpecies` through the `hasSpecies` object property, and to the `WaterArea` through the `hasWaterArea` object property. Figure 2.7 shows the definition of the class `AquaticResource` in the fao-application ontology.
Although this path can lead to the retrieval of the desired result, it is based on a strong assumption i.e. an aquatic species is present only in one water area and at only one depth. This assumption is likely to be wrong, and it is not expressed by the ontology axiomatization. In fact, if the intended meaning was based on this assumption all the mentioned properties defined in the fao-application ontology, i.e., `bathymetry`, `hasSpecies`, and `hasWaterArea`, should be defined as `owl:FunctionalProperty`.
In this case, the evaluation of the fao-application ontology is negative with respect to the analyzed competency question.



OWL – NeOn Toolkit – /Applications/NeOnToolkit-1.2.2-B904-extended-Mac/NeOn Toolkit.app/Contents/MacOS/workspace

- DUL [OWL]
- codolight [OWL]
- fao [OWL]
  - http://www.fao.org/fsdas/v2.0
  - http://www.ontologydesignpatterns.org/cp/owl/fsdas/spec
    - Classes
      - AquaticSpecies
      - BathymetricRange
      - Habitat
      - situation:Situation
        - SpeciesConditions
      - WaterArea
    - Object Properties
    - Data Properties
    - Annotation Properties
    - Datatypes
    - R2O Mappings
  - http://www.ontologydesignpatterns.org/cp/owl/situation.o
  - http://www.ontologydesignpatterns.org/schemas/cpannota

OWL Class

URI <http://www.ontologydesignpatterns.org/cp/owl/fsdas/speciesconditions.owl#SpeciesConditions>

Super Restrictions

Quantifier	Min/Max	Property	Range
EXACTLY/CARD.	1	forSpecies	owl:Thing
EXACTLY/CARD.	1	hasBathymetricRange	owl:Thing
EXACTLY/CARD.	1	hasHabitat	owl:Thing
EXACTLY/CARD.	1	inWaterArea	owl:Thing

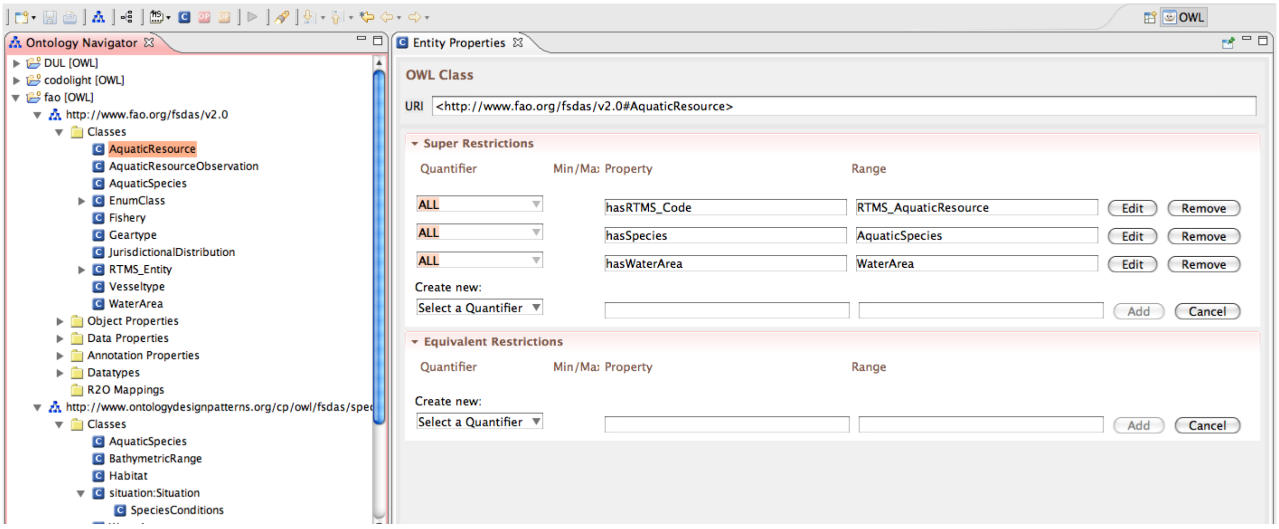
Create new: Select a Quantifier

Equivalent Restrictions

Quantifier	Min/Max	Property	Range

Create new: Select a Quantifier

Figure 2.6: The definition of the class `SpeciesConditions` in the *SpeciesConditions* Content ODP. It can be noticed that the `SpeciesConditions` class is modeled as an nary-relation which relates an `AquaticSpecies` to a `BathymetricRange`, a `WaterArea`, and an `Habitat`.



- DUL [OWL]
- codolight [OWL]
- fao [OWL]
  - http://www.fao.org/fsdas/v2.0
    - Classes
      - AquaticResource
      - AquaticResourceObservation
      - AquaticSpecies
      - EnumClass
      - Fishery
      - Geartype
      - JurisdictionalDistribution
      - RTMS_Entity
      - Vesseltype
      - WaterArea
    - Object Properties
    - Data Properties
    - Annotation Properties
    - Datatypes
    - R2O Mappings
  - http://www.ontologydesignpatterns.org/cp/owl/fsdas/spec
    - Classes
      - AquaticSpecies
      - BathymetricRange
      - Habitat
      - situation:Situation
        - SpeciesConditions

OWL Class

URI <http://www.fao.org/fsdas/v2.0#AquaticResource>

Super Restrictions

Quantifier	Min/Max	Property	Range
ALL		hasRTMS_Code	RTMS_AquaticResource
ALL		hasSpecies	AquaticSpecies
ALL		hasWaterArea	WaterArea

Create new: Select a Quantifier

Equivalent Restrictions

Quantifier	Min/Max	Property	Range

Create new: Select a Quantifier

Figure 2.7: The definition of the class `AquaticResources` in the fao-application ontology.

Chapter 3

Evaluation WikiFlow

This chapter presents *Evaluation WikiFlow*, a wiki-based tool supporting workflows for the evaluation of knowledge objects e.g., ontologies, vocabularies, articles, ontology design patterns..
In order to support the XD approach we have started the ontologydesignpatterns.org initiative[1] (odp initiative) that has been already introduced in Deliverable 2.5.1 [PGD$^+$08]. Based on requirements coming from the odp initiative scenario, we have implemented a tool which supports the collaborative evaluation of knowledge objects, including ontology design patterns. Such a tool, named *Evaluation WikiFlow* (described in section 3), currently runs on the odp initiative web site where it is used for supporting the evaluation of ontology design patterns. Evaluation WikiFlow does not commit to a specific method, hence it can be customized in order to support different methods e.g. EXD, Open Rating.
Evaluation WikiFlow is described in the context of its use in the odp initiative. In this way, we also describe structure and main aspects and features of the odp initiative (Section 3.1).

Evaluation WikiFlow has been released in alpha version as open source software and can be downloaded from the MediaWiki wiki site[2]. The reader can test it on ontologydesignpatterns.org, where it is associated to `ProposedCP` articles.
Evaluation WikiFlow is designed in order to store a semantic representation of the evaluation history of an article. This feature is motivated by the goal of semantically representing the rationales behind an evaluation so as to identify recurrent mistakes and good practices (of ontology design in the case of odp initiative).
Evaluation WikiFlow is substantiated by a tab, labelled as *evaluation*, which is added on top of the wiki page, as shown in Figure 3.1.

In order to describe Evaluation WikiFlow we refer to the case of its application in case of the ontology designpatterns.org initiative, which is a NeOn result as well and that was introduced in [PGD$^+$08]. As from that time, further development have been done, we describe the current status and structure of the web site with the twofold purpose of providing an update and to better contextualize the scenario we use for describing Evaluation WikiFlow.
The chapter is organized as follows: in Section 3.1, we briefly introduce the ontologydesignpatterns.org initiative, in section 3.2 we describe the Evaluation WikiFlow (EWF) tool, in Section 3.3 some related work are discussed. Finally, Section 3.4 discusses some remarks and future development.

3.1 The ontologydesignpatterns.org initiative

Ontologydesignpatterns.org (odp) is a semantic web portal dedicated to ontology design best practices for the semantic web, with particular focus on ontology design patterns (ODPs). odp is targeted at users who are interested in best practices for ontology design and ontology engineering, and ODPs are reusable solutions to recurrent modeling problems. As such, they are a preferential route for designing high-quality ontologies. In [PG08], ODPs are classified into different types: logical, content, presentation, correspondence, etc.

[1] http://www.ontologydesignpatterns.org
[2] at http://www.mediawiki.org/Extensions:EvalWF

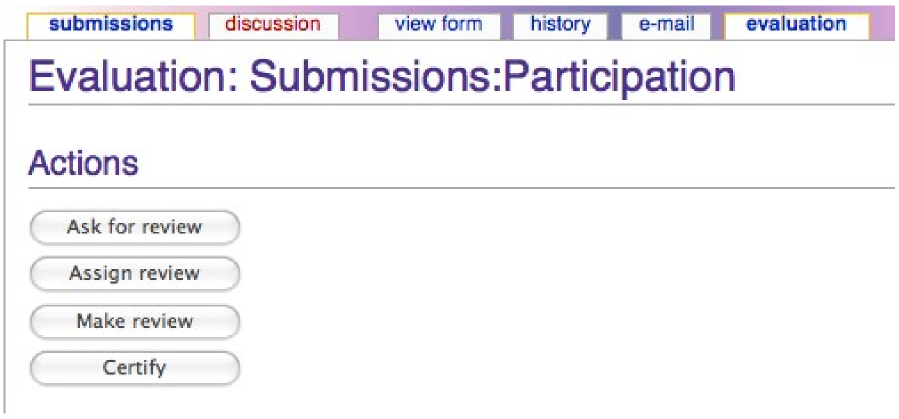


Figure 3.1: The WikiFlow evaluation tab on an ODP page.

Currently, odp supports the lifecycle of *Content* ontology design patterns (Content ODPs) (solutions to content modeling problems, e.g for time, space, biological sequencing, geographic areas, invoicing.). Being integrated with specific modeling problems, Content ODPs are ideally reusable components for building ontologies that are valid with reference to their task and domain, and due to their simplicity, they can be easily exploited as reference vocabularies for publishing linked data on the web. Currently, odp is under improvement for supporting the lifecycle of other ODP types, and it is also used as a resource for enhancing collaborative ontology design with the NeOn Toolkit[3], e.g. by matching, specializing, and composing Content ODPs in specific ontology projects [PGD^{+}08].

ontologydesignpatterns.org Organization

According to the functionality that we have implemented so far, and depicted in Figure 3.2, odp features different areas and types of users. Currently, the web portal is organized into the following areas:

3.1.1 Community

This area is meant to receive odp users' contributions and discussions. It includes a facility for posting modeling issues to the aim of sharing experiences and adopted solutions with the community.

3.1.2 Proposals

This is the area where odp users can propose ODPs[4]. It includes a catalogue of all proposed Content ODPs. There are two alternative procedures for proposing a pattern, both are accessible through the *Propose a Content ODP* functionality.

- the user chooses a name for the Content ODP and is guided by a simple form that, once filled and saved, is posted as a new Content ODP proposal;
- the user uploads the owl file of the Content ODP, and the system automatically fills the Content ODP form, that once accepted by the user is posted as a Content ODP proposal. The form can be further edited by the user by the *edit with form* functionality. In order to exploit this facility at its best, the owl file must be annotated in terms of `rdfs:label`, `rdfs:comment`, and annotation properties defined

[3]http://www.neontoolkit.org
[4]Currently, only Content ODP proposals are supported.

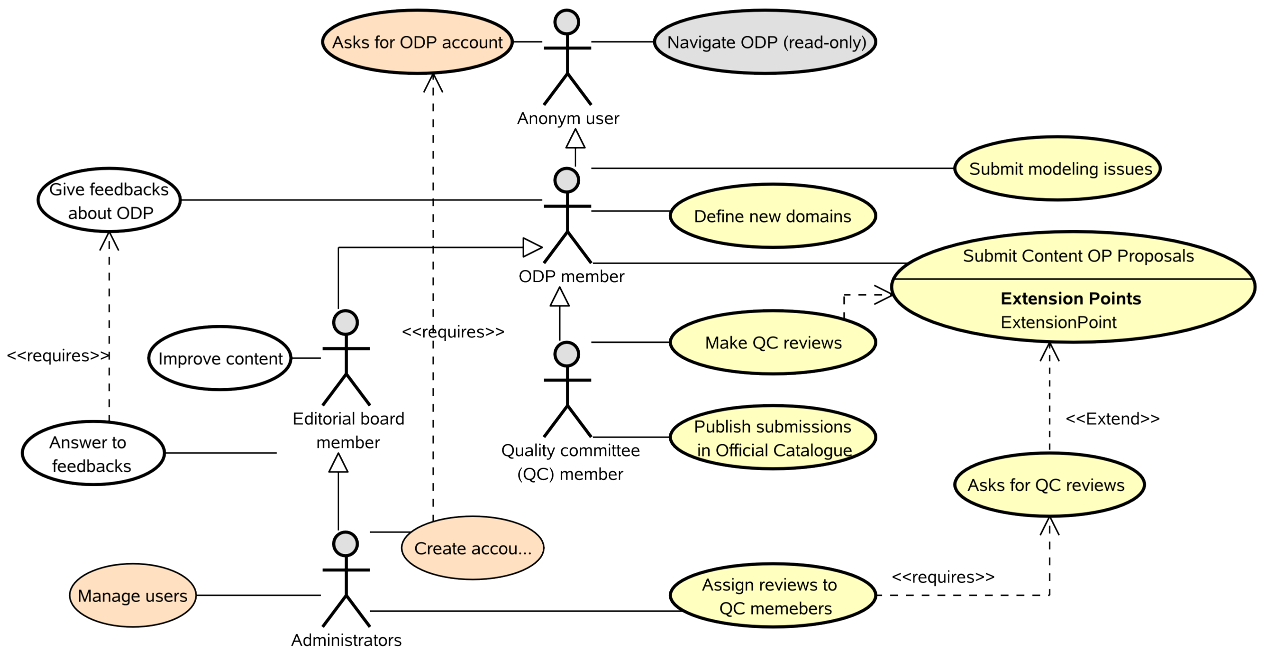


Figure 3.2: odp Use Case Diagram: main functionalities and user roles.

by the *Content ODP annotation schema*[5]. All these annotation properties are mapped to semantic relations into odp.

The proposed ODPs are expected to come from practical and successful experiences of ontology development. All proposed patterns belong to the odp namespace named `Submissions`.

3.1.3 Reviews

This area collects and publishes the reviews of proposed Content ODPs. Proposed Content ODPs can be reviewed by all registered users i.e. *open reviews*. However, they are eventually reviewed by at least two members of the Quality Committee, formed by ontology experts, i.e., *QC reviews*. The aim of reviews is twofold: on one hand, they provide odp users with explicit rationales behind the evaluation of specific domain solutions. On the other hand, reviews provide the author of a certain Content ODP with guidelines for fixing possible problems in order to get the Content ODP certified.
Open reviews are extremely useful to Quality Committee members while formulating their review, and also for identifying new Quality Committee members from the community.

3.1.4 Catalogue

This is the official Content ODP catalogue. This area collects all Content ODPs that are certified by the odp Quality Committee. The only difference between certified and proposed Content ODPs is that the formers are guaranteed to be fully described (with regard to odp specification)[6], validated by the odp Quality Committee, and *always* associated with a reusable OWL implementation available for download that has a stable namespace.

3.1.5 Training

This area collects tutorials, publications, and sample modeling exercises.

[5]http://ontologydesignpatterns.org/schemas/cpannotationschema.owl
[6]See also the cp annotation schema: http://www.ontologydesignpatterns.org/schemas/cpannotationschema.owl

3.1.6 Feedback

This is the area where odp users can post feedback entries. Typically, feedback entries identify issues to be addressed in order to improve odp. The editorial board and the odp core developers treat such feedback as input/suggestions for raising new development and/or maintenance requirements.

3.1.7 Domain

This page lists all domains that have been defined by odp users, and allows them to create new ones. Each Content ODP or Modeling Issue is associated with a domain e.g. Organization, Business, Health, Music, etc., by the semantic property `domain`. This semantic property is extremely useful for users who are looking for Content ODPs for a certain domain of knowledge. In fact, they can query odp about all Content ODPs addressing a certain domain.
Read and write access to odp areas is handled by a policy based on five different types of users. Types of users are: *AnonymousUser*, *ODPUser*, *EBMember*, *QCMember*, *Administrator*.

- **Anonymous users** *(AnonymousUser)*. They have read-only access to the whole portal, and can request an account.
- **odp users** *(ODPUser)*. This user type represents everybody who has registered to odp through the account request page, and hence it is part of the odp Community. Such users can create and/or edit articles in the Community area, post modeling issues, create and/or edit pages about domains, submit proposals, post feedback entries, make open reviews about proposals, and participate in the discussions (through the discussion page).
- **Editorial board** *(EBMember)*. Users in this group answer feedback, propose and contribute to content improvement through a page dedicated to development tasks[7]. The Feedback area is used by editorial board members in order to identify possible development tasks. The development task page is directly used by core developers, and the editorial board members are entitled to manage task priority.
- **Quality committee members** *(QCMember)*. These users are entitled to posting *QC reviews* about proposals. Certification of Content ODP that are published in the official catalogue is based on QC reviews, and they are handled in a peer-reviewing like approach. The QC evaluation workflow is managed by the Evaluation WikiFlow extension.
- **Administrators** *(Administrator)*. Users in this group manage account creation, odp technical issues and upgrades.

Content ODP presentation

The odp software is based on Media Wiki[8], Semantic Media Wiki (SMW)[9], Semantic Forms (SF)[10], and other extensions[11]. We have exploited such extensions' features in order to semantically represent as much knowledge as possible about the ontology design patterns. All wiki articles are assigned with a category, and can be related to each others through semantic relations. The core of odp is made of articles that represent ontology design patterns (currently only Content ODPs). According to [PG08] we have defined a set of semantic properties that comply to the *Content ODP annotation schema*[12]. They allow us to specify useful semantic information about Content ODPs, and have guided us as well in defining the visualization template

[7] http://ontologydesignpatterns.org/index.php/Odp:Development
[8] http://www.mediawiki.org
[9] http://www.semantic-mediawiki.org
[10] http://www.mediawiki.org/Extension:SemanticForm
[11] The full list of the extensions can be found at http://ontologydesignpatterns.org/index.php/Special:Version
[12] http://www.ontologydesignpatterns.org/schemas/cpannotationschema.owl

of Content ODP pages.
Consider for example Figure 3.3. It depicts the odp page of the Content ODP named *situation*, which is in the catalogue of proposals[13]. Under the diagrammatic representation of the Content ODP (which is a UML class diagram automatically obtained by Top Braid Composer[14]) on the center-right side of the page there is a box containing Content ODP properties. They include: the person who submitted the proposal; the name of the Content ODP; its intent, consequences, and typically associated competency questions; relations to other Content ODPs, such as its components, its specializing Content ODPs, and Content ODPs that are specialized by it; the URL of the downloadable OWL building block, the source from which it has been extracted, and so on. On the center-left side of the page, all ontology elements defined in the Content ODP are listed and described. Furthermore, each of them has its own wiki article of category `OntologyElement`. On the bottom of the page, there are links to examples of scenarios modeled by using the Content ODP, such examples includes a UML object diagram, a natural language description, and a link to the OWL file that implements it.
Finally, on the very bottom of the page, there are links to possible reviews of the patterns.
Next section describes Evaluation WikiFlow, a SMW extension that is able to support workflows for the evaluation of wiki articles.

3.2 Evaluation WikiFlow tool

Evaluation WikiFlow's features can be described from two perspectives: configuration and functionality. Configuration features include the following ones;

Selection of article's categories that have to be associated with the evaluation tab. This feature allows users to activate the evaluation tab and its functionalities (see later in this section) for a set of categories defined by the ontology behind the wiki, e.g. currently odp activates it for the `ProposedCP` category.

Customization of the semantic form associated to the review page. This feature allows users to define the review schema associated with the correspondent semantic form.

Definition of different semantic review forms associated to different article's categories. This feature allows users to associate different categories (or sets of categories) with semantic forms having different review schemas E.g. in odp, Logical ODPs will be reviewed by using a form different from the one used for reviewing Content ODPs.

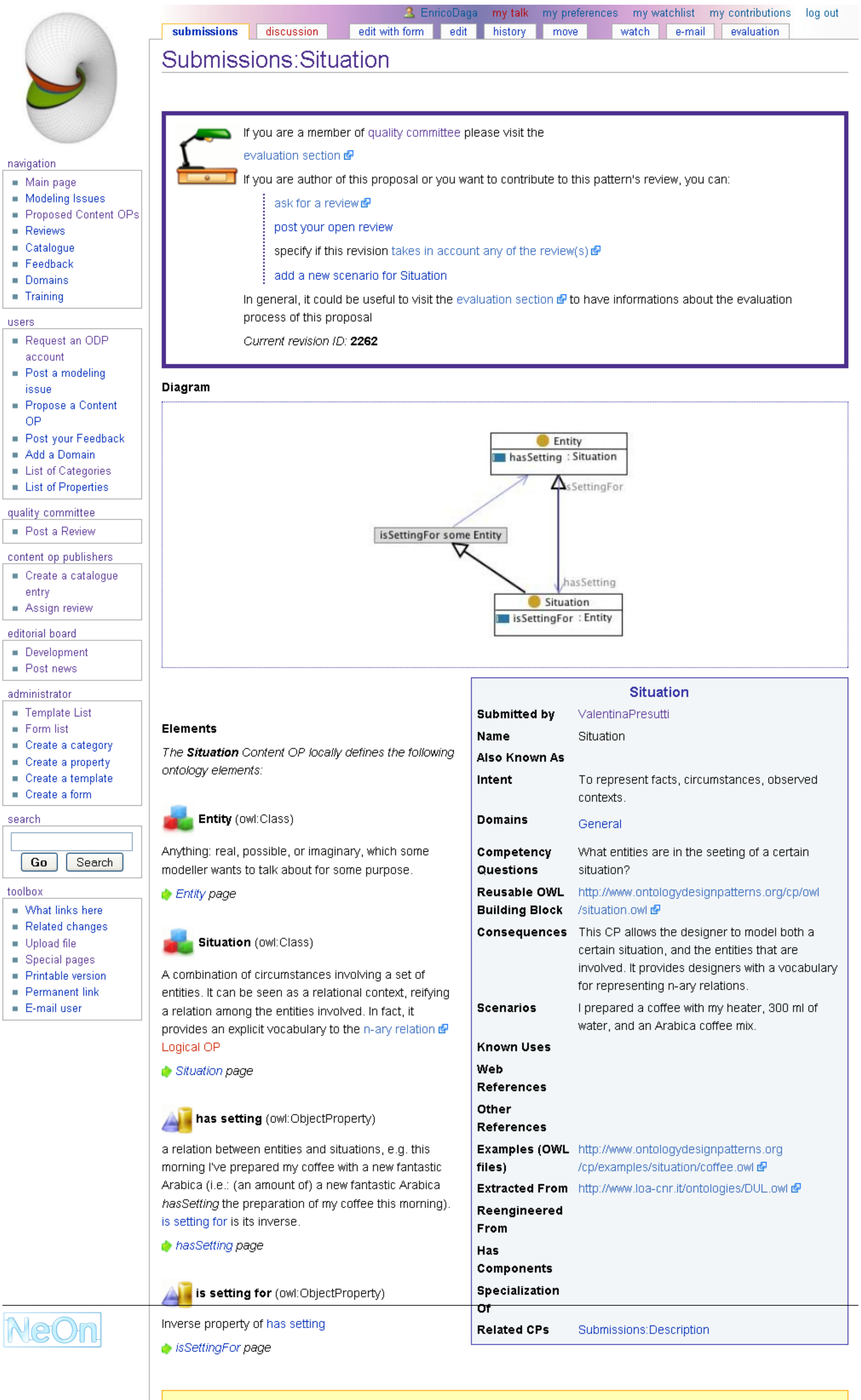
Access rights configuration. Evaluation WikiFlow adds five new permissions to the existing ones. They can be configured in order to allow/prevent users to certify articles, and to perform the following actions on reviews: view, ask for, assign, and make. E.g. odp *QCMember* users have the rights for making a review, while every *ODPUser* can ask for a review.

From the functionality side, Evaluation WikiFlow provides the following features, accessible from the evaluation tab (see Figure 3.1):

Ask for Review. This functionality allows users to ask for a review of the current article. Once this action has been performed, the article is automatically assigned to the category `WaitingForReview` that represents the current state of the article.

Assign Review. This functionality allows users to commit another user with the task of reviewing an article. E.g. the odp *EBMember* users are allowed to perform this action in order to assign the reviewing of a Content ODP proposal to some *QCMember* users.

[13] http://ontologydesignpatterns.org/index.php/Submissions:Situation
[14] http://www.topquadrant.com/topbraid/composer/index.html



EnricoDaga my talk my preferences my watchlist my contributions log out

submissions | discussion | edit with form | edit | history | move | watch | e-mail | evaluation

navigation
- Main page
- Modeling Issues
- Proposed Content OPs
- Reviews
- Catalogue
- Feedback
- Domains
- Training

users
- Request an ODP account
- Post a modeling issue
- Propose a Content OP
- Post your Feedback
- Add a Domain
- List of Categories
- List of Properties

quality committee
- Post a Review

content op publishers
- Create a catalogue entry
- Assign review

editorial board
- Development
- Post news

administrator
- Template List
- Form list
- Create a category
- Create a property
- Create a template
- Create a form

search

Go Search

toolbox
- What links here
- Related changes
- Upload file
- Special pages
- Printable version
- Permanent link
- E-mail user

Submissions:Situation

If you are a member of quality committee please visit the evaluation section

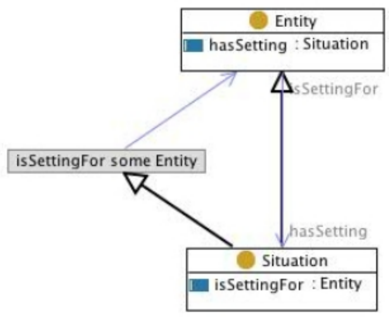
If you are author of this proposal or you want to contribute to this pattern's review, you can:

- ask for a review
- post your open review
- specify if this revision takes in account any of the review(s)
- add a new scenario for Situation

In general, it could be useful to visit the evaluation section to have informations about the evaluation process of this proposal

Current revision ID: **2262**

Diagram



Elements

*The **Situation** Content OP locally defines the following ontology elements:*

Entity (owl:Class)

Anything: real, possible, or imaginary, which some modeller wants to talk about for some purpose.

Entity page

Situation (owl:Class)

A combination of circumstances involving a set of entities. It can be seen as a relational context, reifying a relation among the entities involved. In fact, it provides an explicit vocabulary to the n-ary relation Logical OP

Situation page

has setting (owl:ObjectProperty)

a relation between entities and situations, e.g. this morning I've prepared my coffee with a new fantastic Arabica (i.e.: (an amount of) a new fantastic Arabica *hasSetting* the preparation of my coffee this morning). is setting for is its inverse.

hasSetting page

is setting for (owl:ObjectProperty)

Inverse property of has setting

isSettingFor page

Situation

Submitted by	ValentinaPresutti
Name	Situation
Also Known As	
Intent	To represent facts, circumstances, observed contexts.
Domains	General
Competency Questions	What entities are in the seeting of a certain situation?
Reusable OWL Building Block	http://www.ontologydesignpatterns.org/cp/owl/situation.owl
Consequences	This CP allows the designer to model both a certain situation, and the entities that are involved. It provides designers with a vocabulary for representing n-ary relations.
Scenarios	I prepared a coffee with my heater, 300 ml of water, and an Arabica coffee mix.
Known Uses	
Web References	
Other References	
Examples (OWL files)	http://www.ontologydesignpatterns.org/cp/examples/situation/coffee.owl
Extracted From	http://www.loa-cnr.it/ontologies/DUL.owl
Reengineered From	
Has Components	
Specialization Of	
Related CPs	Submissions:Description

Make Review. By performing this action, the reviewer faces a review form and can post a review. A new article containing the review is created and named by concatenating the user name of the reviewer and the name of the article under evaluation. If the article under evaluation has the `WaitingForReview` category, this is updated by removing it, e.g., odp *QCMember* users can perform this action in odp. A possible future enhancement of this feature would be to make the system aware of additional status of an article associated to an equal number of actions e.g., when an article has received all expected reviews it enters a `WaitingForRevision` status and th system asks the authors to revise the article according to all reviews.

Certify. This functionality allows users to certify an article. Once an article is certified it is *freezed* i.e. it cannot be modified anymore. Furthermore, the evaluation tab of the article reports its evaluation history (see next bullet). The user has the possibility to automatically create a copy of the article. The copy will have a completely separated lifecycle from the previous one e.g. a copy of a certified Content ODP proposal is created in the odp `Catalogue` namespace and can be modified in order to indicate a persistent identifier for the building block, etc.

Semantic report of evaluation history. The evaluation tab of an article is dynamically updated with its evaluation history. Each review of the article is associated with two article's versions: the one under evaluation and the possible updated version of the article that takes into account the review. The editor of the article can express the association "takes into account" between the revised article and the addressed review, by means of a "one click" functionality. The semantic report of the evaluation history exploits the MediaWiki versioning system. In the context of odp, our aim is to use the evaluation history as a source for analyzing the rationales behind the evaluation of patterns as well as for extracting good practices and common mistakes, e.g. anti-patterns [PGD$^+$08] in ontology design.

In order to illustrate the Evaluation WikiFlow extension, we show (see Figure 3.4) its instantiation in the odp case. The evaluation tab is used in order to support the certification process of patterns. After posting a Content ODP proposal, an odp user can submit it to the certification process by asking a review, i.e., by clicking in the `ask for review` button in the evaluation tab. The Content ODP proposal is added to the list of Content ODPs that are waiting for reviews (the category `WaitingForReview` is assigned to the Content ODP). Such list is handled by the editorial board.
An *EBMember* user eventually assigns the reviewing task to at least two *QCMember* users. When a review is created, the category of the Content ODP proposal is updated, i.e., `WaitingForReview` is removed from the list of the Content ODP's categories, and a link to the review is added in the Content ODP proposal page.
The author of the Content ODP can modify it in order to address the review issues. Once the author believes that a certain version of the Content ODP addresses one or more reviews, (s)he can associate such version to them with the relation `takesIntoAccountReviews`[15].
This process is iterated until reviewers agree on the certification of the Content ODP. Certified Content ODPs are copied in the official catalogue, i.e., the `Catalogue` namespace.

3.3 Related work

In the past few years, there has been lots of research and development effort on good practices of web ontology design. Literature spans from reports of the W3C OEP task force [PG04, NA05], to methodology- and ontology design patterns-oriented books and scientific papers e.g., [PG08, PST04]. Additionally, repositories of reusable components are available on the web[16]. In the context of Semantic Web research and applications, ontology design patterns (ODPs) are now a hot topic. They have been introduced by [GCB04, RR04, Sva04]. Some work [Blo05] has also attempted a learning approach (by using case-based reasoning) to derive and

[15]This is a "one click" operation.
[16]E.g. http://odps.sourceforge.net/

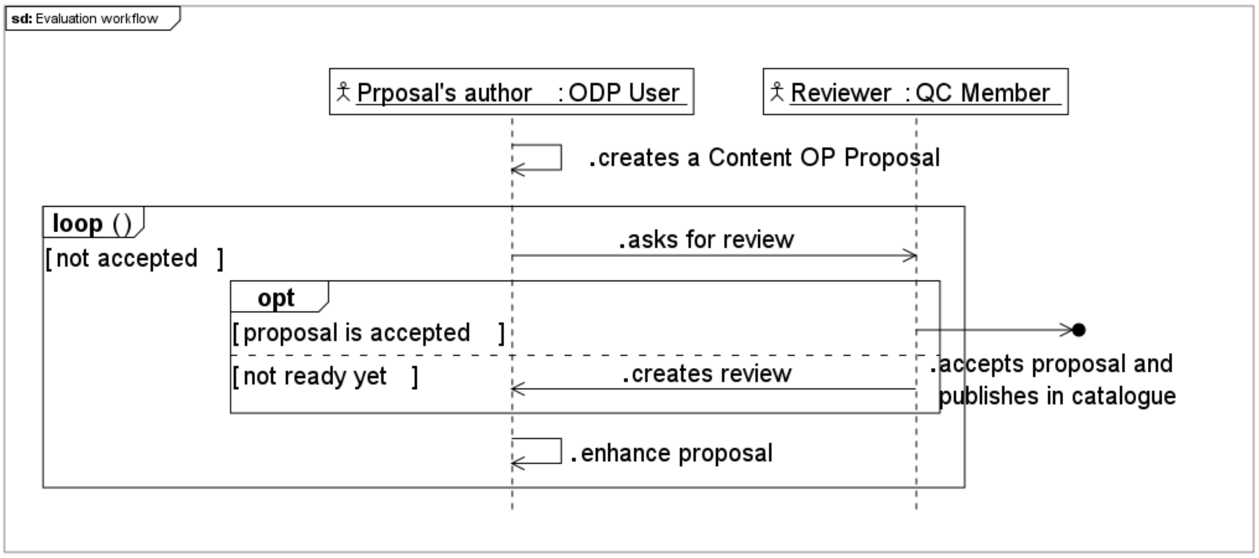


Figure 3.4: Evaluation example

rank patterns with respect to user requirements. The research has also addressed domain-oriented patterns, e.g. for content objects and multimedia [ATS$^+$07], software components [Obe06], business modelling and interaction [GP07, GW04], relevance [Jua07].
For an historical perspective and a more detailed survey, the reader can refer to [Gan05, GP07, Hay96, Gui05].
odp is about ontology design patterns in the sense of [Gan05] and [PG08]. It is meant to provide semantic web users with a community-based web portal for training and discussion, associated with a semantic web repository of ontology design patterns. Additionally, evolving requirements of odp has helped us in developing new functionalities that we release as domain-independent SMW extensions, such as Evaluation WikiFlow, presented here. Wiki-based workflow support is also provided by other existing approaches. For example, a peer-review system has been set for specific works over wikipedia which are going to meet high quality standards[17], in this case the process is not based on semantic features and is not completely automated i.e. the system does not activate actions with respect to certain status, and the focus is not on semantic tracing of rationales. Another tool that supports wiki-based workflows, even if it is not specifically designed for evaluation, is the CICERO SMW extension [DEB$^+$08]. It is designed for supporting argumentation based on the argumentation part of the DILIGENT [PST04] methodology. CICERO might be used as a complementary tool to Evaluation WikiFlow, e.g., in case one wants to use the DILIGENT argumentation model in order to support interaction between reviewers, i.e., before certify actions in Evaluation WikiFlow.

3.4 Conclusion and future work

In this chapter, we have described ontologydesignpatterns.org (odp), a semantic web portal supporting the community around best practices of ontology design for the Semantic Web, with particular focus on ontology design patterns (ODPs). Starting from odp requirements of supporting the certification process of patterns, we have developed a SMW extension for supporting the process of wiki articles' evaluation, also presented here and named Evaluation WikiFlow. We have planned a user-based evaluation by monitoring the odp community activity. odp ongoing and planned work includes:

- support for new types of ontology design patterns, according to the ontology pattern classification described in [PG08];

[17] http://en.wikipedia.org/wiki/Wikipedia:Peer_review

- a keyword-based search service based on Watson [Wat];
- the odp repository APIs that will allow to query and select the odp repository of patterns;
- an OWL/RDF export service;
- augmenting Content ODPs by means of Linking Open Data[18];
- an open rating system for open reviews based on [Lew06];
- statistical monitoring of Content ODP downloads to be used as a dimension of user-based evaluation of Content ODPs and odp usage;
- a user-based evaluation of Evaluation WikiFlow based on the odp certification process.

For additional information on ongoing work on ODP, the reader can refer to the development task page[19].

[18] http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData
[19] http://ontologydesignpatterns.org/index.php/Odp:Development

Part II

Trust-based Evaluation of Ontology Components

Chapter 4

Implementation and Evaluation of the TS-ORS

After providing the theoretical details of the topic-specific open rating system (TS-ORS) in D2.2.1 [SAd$^+$07], this section deals with a concrete implementation of the system[1]. The concrete implementation described hereafter is part of the integration of the TS-ORS and the Open University's Semantic Web gateway Watson, and thus was implemented with concrete application requirements in mind.

The TS-ORS allows ranking of ontologies based on user reviews and the trust that users express in these reviews. Everybody can share their opinion about an ontology–property combination and these reviews will be displayed to other users. A review consists of a 5-star rating and a justification for that rating. Other users can then state whether they find the review helpful or not and thus express trust to the reviewers. Based on the trust information, a web of trust is computed that allows the ranking of reviews for each ontology–property combination. It allows a personalized result for users that are logged in, and a ranking based on common opinion in the other case. Knowing which reviews are best for each ontology–property combination, a linear combination of the different properties allows to compute an overall rating for the ontology. Using these scores, the ontologies can be ranked.

The Watson system[2] can be seen as a search engine for the Semantic Web. It supports users and applications in finding, selecting and (re)using ontologies that are available online, through advanced search and exploration mechanisms.

Combining the TS-ORS and Watson allows users of Watson to directly retrieve quality information on ontologies stored in Watson (in case somebody reviewed that ontology). A common problem when reusing ontologies or parts of ontologies, is that the quality is unknown. Because the number of ontologies one could possibly reuse may be large, it is impossible to assess all ontologies by yourself. For that reason information from the community is needed. Furthermore, the integration allows Watson users to influence the ranking of results by changing parameters and expressing trust.

Based on the scenario described above some concrete design decisions were made, which will be described in the following section. This chapter is structured as follows: First design decisions are explained, then the overall architecture and concrete implementation details are presented. Lastly the results of a performance evaluation are discussed.

4.1 Design Decisions

The functionality needed for Watson was basically storing and retrieving reviews and providing ranking and rating results. Furthermore, metatrust statements are supported, but only on the level of ontologies, users and properties, since Watson does not support domain information yet. This feature can however easily

[1]SourceCode is available on request by contacting `lewen'@'aifb.uni-karlsruhe.de`
[2]`http://watson.kmi.open.ac.uk`

Table 4.1: Metatrust Statements Supported Currently in Implementation

Statement	**Scope**	**Explanation**
W	$A_i \times A_j \times O_n \times X_k \times D^{O_n} \rightarrow T_u$	Statement on a specific property of a specific ontology
$W_{O_n X}$	$A_i \times A_j \times O_n \rightarrow T_u$	Statement on all properties of a specific ontology
W_{CX_k}	$A_i \times A_j \times X_k \rightarrow T_u$	Statement on a specific property of all ontologies
W_{CX}	$A_i \times A_j \rightarrow T_u$	Statement on all properties of all ontologies

be added at a later point, since it only affects the metatrust propagation. So in terms of trust statements mentioned in [SAd+07], only those displayed in Table 4.1 are allowed in this concrete implementation. In terms of interaction, the TS-ORS takes care of assigning the internal IDs needed for computation and storage (see Section 4.2.1). It takes as input URIs as identifiers for ontologies and is flexible in terms of user identification. It has its own MySQL database to store reviews, user, ontology and trust information. The application is implemented in JAVA, and servlets are used for interacting with Watson and other programs. At the moment, REST services are offered that provide the results either as JSON, XML or HTML, based on content negotiation using the HTTP header.

4.2 Architecture

The first decision taken was how the database schema should look like for optimal retrieval and partitioning of information.

4.2.1 Database Schema

One of the foremost concerns when using databases in an application is developing a database schema that ensures data integrity but does not sacrifice performance. In order to optimize performance, it is necessary to use database indexes and ensure sufficient memory is allocated for caching. In our case we decided to store the most basic information, like user, ontology properties, ontologies, ratings and trust between users and metatrust in dedicated tables (see also Figure. 4.1). We chose MySQL as a database with MyISAM storage engine for better performance. We will now describe the different tables:

- **Users**: This table stores the relation between the user ID (uid) and user data.
- **Ontologies**: This table stores the relation between the ontology ID (oid) and the ontology uri.
- **Properties**: This table stores the relation between the property ID (xid) and the name of that property.
- **Rating**: This table stores the relation between the rating ID (rid) and the review. For each review, a combination of uid, oid and xid exists along with the review data $dext$ (textual part of the review) and $dstar$ (the 5 star value).
- **Trust**: stores the relation between the trust ID (tid) and the trust statement. For each trust statement, a combination of uid, rid exists along with the information whether trust or distrust was expressed.
- **Metatrust**: stores the relation between the metatrust ID ($mtid$) and the metatrust statement. Three different metatrust statements are possible (also see Table 4.1). Depending on the type of metatrust statement, $global, ontology$ or $property$ is set to 1 in case of metatrust and -1 in case of metadistrust, and the needed information is stored in the table.

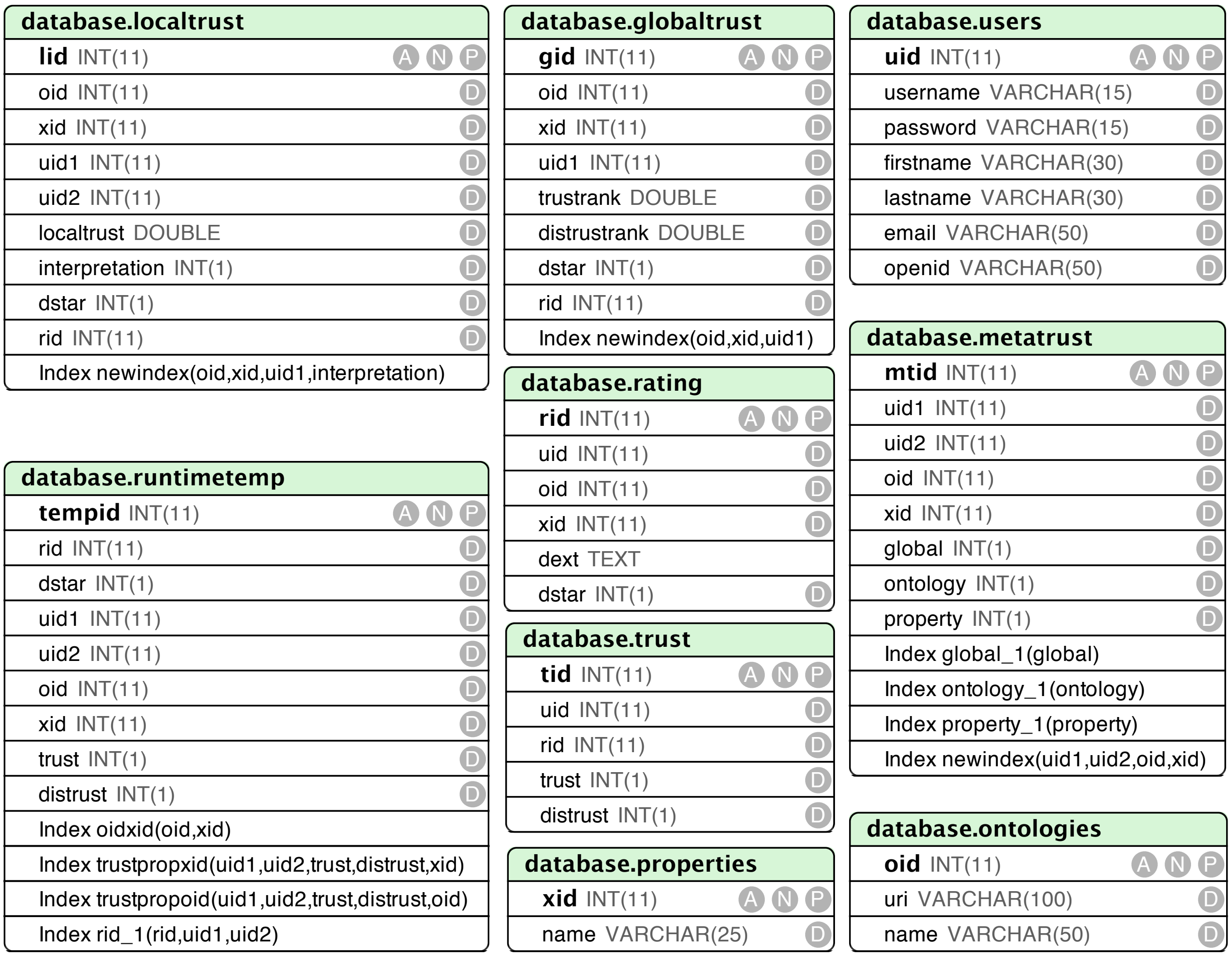


Figure 4.1: This graphic depicts the database schema used for the implementation of the TS-ORS.

- **Runtimetemp**: During the initial computation of $globaltrust$ and $localtrust$, a table called temp is temporarily created. This table becomes runtimetemp after the computations are completed. The only reason for having the table is optimizing performance. The table is basically the result of joining the trust and rating table. This materialization saves time because specific indexes can be created and the join only has to be performed once. As long as there are not two separate databases for runtime data and offline data, the database will be in use when the recomputation of trust is triggered. So during creation, the fresh data has to be stored in temporary tables that can later replace the current runtime tables.

- **Globaltrust**: This table stores the trustrank and distrustrank of users for a given oid, xid combination. To improve runtime-performance, also the dstar and rid of the review are stored in this table. During the initial computation, a table tempglobaltrust with identical schema is used, which becomes globaltrust once the computations are complete. The need for a runtime and a temporary table is also due to the fact that otherwise small updates could not be directly displayed without recomputing everything.

- **localtrust**: This tables stores the localtrust value along with its interpretation as trust or distrust for a given oid, xid combination. To improve runtimeperformance, dstar and rid of the reviews are also stored in the table. During the initial computation, a table templocaltrust with identical schema is used, which becomes localtrust after the computations are complete.

4.2.2 UML-Diagram

The methods and functionality were partitioned into different classes. We will first describe the classes and methods of the core package and then the classes needed for the interaction with external applications.

TS-ORS Core

As can be seen in Figures 4.2 and 4.3, the functionality is distributed among different classes. They are grouped roughly by their functionality. The classes used for the most basic databank interaction are not displayed, since they only provide basic functionality like asking SQL queries and performing SQL updates or getting a connection from the connection pool. We will now quickly describe the different classes. If a method contains the string “update” it means that it performs the changes as well on the runtime databases, so the result of the operation is usually visible immediately instead of only after recomputation of trust values.

- **DBInteraction**: This class contains methods that primarily read and write information from the database. Most methods simply encapsulate SQL queries or update statements. Some variables are used to strore results in between method calls. The behavior of most methods can be inferred from their signature. It is used by methods from other classes for retrieving the needed information from the database and writing the results back.

- **Computations**: This class contains methods that perform the computations needed for determining trustrank, distrustrank, local Trust as well as ranking order and overall rating computation. It also contains methods for performing the majority rounding that has to be performed as part of the local trust computation.

- **Multithreading**: In order to distribute the computation of localtrust and globaltrust (trustrank and distrustrank) among different threads, this class contains all the methods needed for performing the computations and a *run* method to invoke the thread.

- **Metatrust**: This class contains methods that are needed to set and retrieve metatrust statements (see Table 4.1). The method *setEigentrust* is used to ensure that every reviewer trusts his reviews. The *propagateMetaTrustandDistrust* method distributes the metatrust statements down to the level of normal trust statements.

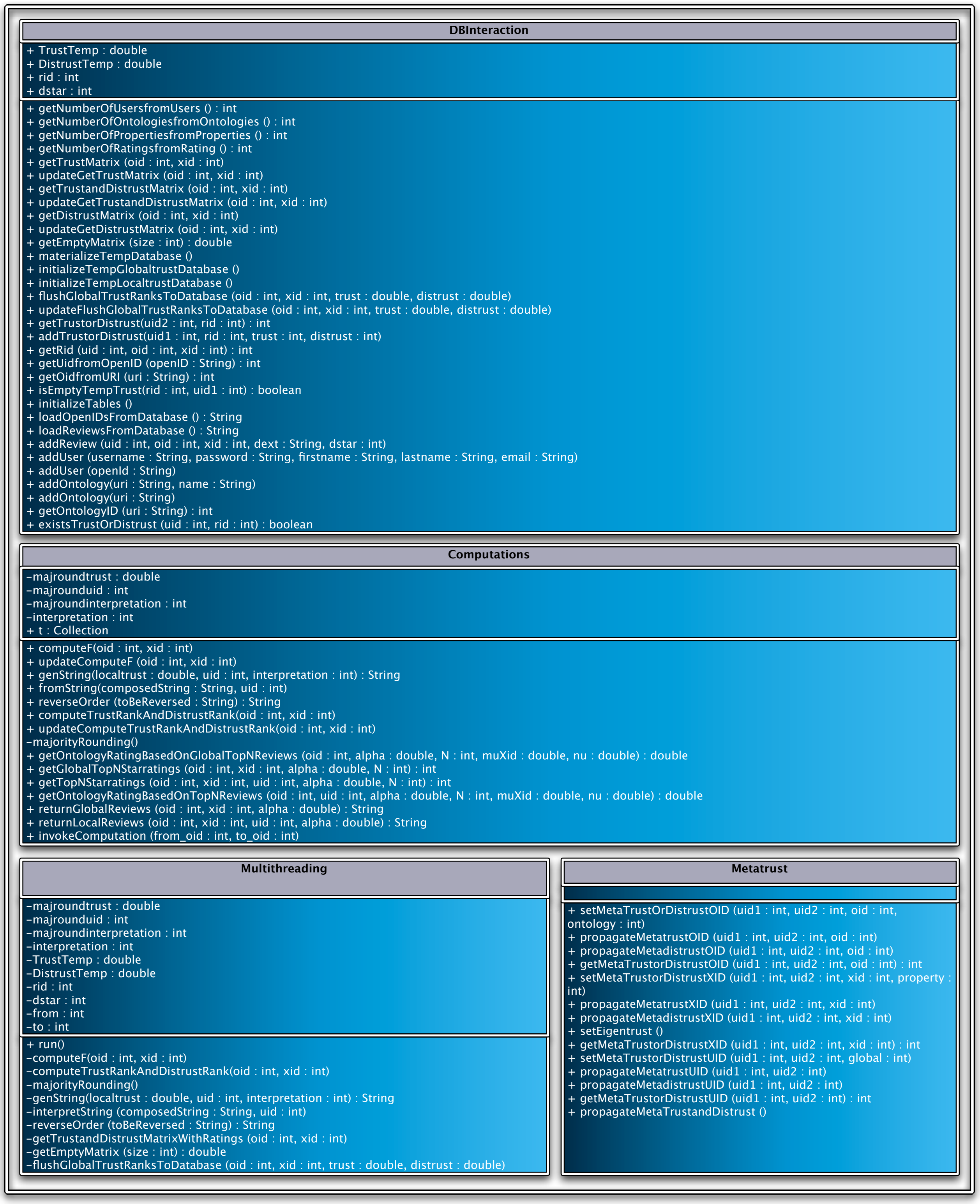
- **Setup**: This class contains methods that can be used to create and drop database tables and indexes.

- **Settings**: This class contains variables that are read by different methods. It is the main place to change settings of the application.

- **Updates**: This class contains methods that are used to alter and refresh data in the system. The method *recomputeEverything* is the main method that refreshes runtime-data based on the information of the database tables. Since not all changes can be directly computed at runtime, it is important to call this method at fixed timepoints (when exactly can be based on size of the system and frequency of changes).

- **GenerateBulkData**: This class contains methods used to generate random data that can be used to test the functionality of the system. Based on the parameters entered, the respective number of instances are created in the database. It is ensured that no duplicates are created.

- **CachedObjects**: This class contains objects that cache information from the database, so that database interaction can be minimized.



```
DBInteraction
+ TrustTemp : double
+ DistrustTemp : double
+ rid : int
+ dstar : int
+ getNumberOfUsersfromUsers () : int
+ getNumberOfOntologiesfromOntologies () : int
+ getNumberOfPropertiesfromProperties () : int
+ getNumberOfRatingsfromRating () : int
+ getTrustMatrix (oid : int, xid : int)
+ updateGetTrustMatrix (oid : int, xid : int)
+ getTrustandDistrustMatrix (oid : int, xid : int)
+ updateGetTrustandDistrustMatrix (oid : int, xid : int)
+ getDistrustMatrix (oid : int, xid : int)
+ updateGetDistrustMatrix (oid : int, xid : int)
+ getEmptyMatrix (size : int) : double
+ materializeTempDatabase ()
+ initializeTempGlobaltrustDatabase ()
+ initializeTempLocaltrustDatabase ()
+ flushGlobalTrustRanksToDatabase (oid : int, xid : int, trust : double, distrust : double)
+ updateFlushGlobalTrustRanksToDatabase (oid : int, xid : int, trust : double, distrust : double)
+ getTrustorDistrust(uid2 : int, rid : int) : int
+ addTrustorDistrust(uid1 : int, rid : int, trust : int, distrust : int)
+ getRid (uid : int, oid : int, xid : int) : int
+ getUidfromOpenID (openID : String) : int
+ getOidfromURI (uri : String) : int
+ isEmptyTempTrust(rid : int, uid1 : int) : boolean
+ initializeTables ()
+ loadOpenIDsFromDatabase () : String
+ loadReviewsFromDatabase () : String
+ addReview (uid : int, oid : int, xid : int, dext : String, dstar : int)
+ addUser (username : String, password : String, firstname : String, lastname : String, email : String)
+ addUser (openId : String)
+ addOntology(uri : String, name : String)
+ addOntology(uri : String)
+ getOntologyID (uri : String) : int
+ existsTrustOrDistrust (uid : int, rid : int) : boolean

Computations
-majroundtrust : double
-majrounduid : int
-majroundinterpretation : int
-interpretation : int
+ t : Collection
+ computeF(oid : int, xid : int)
+ updateComputeF (oid : int, xid : int)
+ genString(localtrust : double, uid : int, interpretation : int) : String
+ fromString(composedString : String, uid : int)
+ reverseOrder (toBeReversed : String) : String
+ computeTrustRankAndDistrustRank(oid : int, xid : int)
+ updateComputeTrustRankAndDistrustRank(oid : int, xid : int)
-majorityRounding()
+ getOntologyRatingBasedOnGlobalTopNReviews (oid : int, alpha : double, N : int, muXid : double, nu : double) : double
+ getGlobalTopNStarratings (oid : int, xid : int, alpha : double, N : int) : int
+ getTopNStarratings (oid : int, xid : int, uid : int, alpha : double, N : int) : int
+ getOntologyRatingBasedOnTopNReviews (oid : int, uid : int, alpha : double, N : int, muXid : double, nu : double) : double
+ returnGlobalReviews (oid : int, xid : int, alpha : double) : String
+ returnLocalReviews (oid : int, xid : int, uid : int, alpha : double) : String
+ invokeComputation (from_oid : int, to_oid : int)

Multithreading
-majroundtrust : double
-majrounduid : int
-majroundinterpretation : int
-interpretation : int
-TrustTemp : double
-DistrustTemp : double
-rid : int
-dstar : int
-from : int
-to : int
+ run()
-computeF(oid : int, xid : int)
-computeTrustRankAndDistrustRank(oid : int, xid : int)
-majorityRounding()
-genString(localtrust : double, uid : int, interpretation : int) : String
-interpretString (composedString : String, uid : int)
-reverseOrder (toBeReversed : String) : String
-getTrustandDistrustMatrixWithRatings (oid : int, xid : int)
-getEmptyMatrix (size : int) : double
-flushGlobalTrustRanksToDatabase (oid : int, xid : int, trust : double, distrust : double)

Metatrust
+ setMetaTrustOrDistrustOID (uid1 : int, uid2 : int, oid : int,
ontology : int)
+ propagateMetatrustOID (uid1 : int, uid2 : int, oid : int)
+ propagateMetadistrustOID (uid1 : int, uid2 : int, oid : int)
+ getMetaTrustorDistrustOID (uid1 : int, uid2 : int, oid : int) : int
+ setMetaTrustorDistrustXID (uid1 : int, uid2 : int, xid : int, property :
int)
+ propagateMetatrustXID (uid1 : int, uid2 : int, xid : int)
+ propagateMetadistrustXID (uid1 : int, uid2 : int, xid : int)
+ setEigentrust ()
+ getMetaTrustorDistrustXID (uid1 : int, uid2 : int, xid : int) : int
+ setMetaTrustorDistrustUID (uid1 : int, uid2 : int, global : int)
+ propagateMetatrustUID (uid1 : int, uid2 : int)
+ propagateMetadistrustUID (uid1 : int, uid2 : int)
+ getMetaTrustorDistrustUID (uid1 : int, uid2 : int) : int
+ propagateMetaTrustandDistrust ()
```

Figure 4.2: This graphic depicts a UML class diagram of some of the TS-ORS core components.

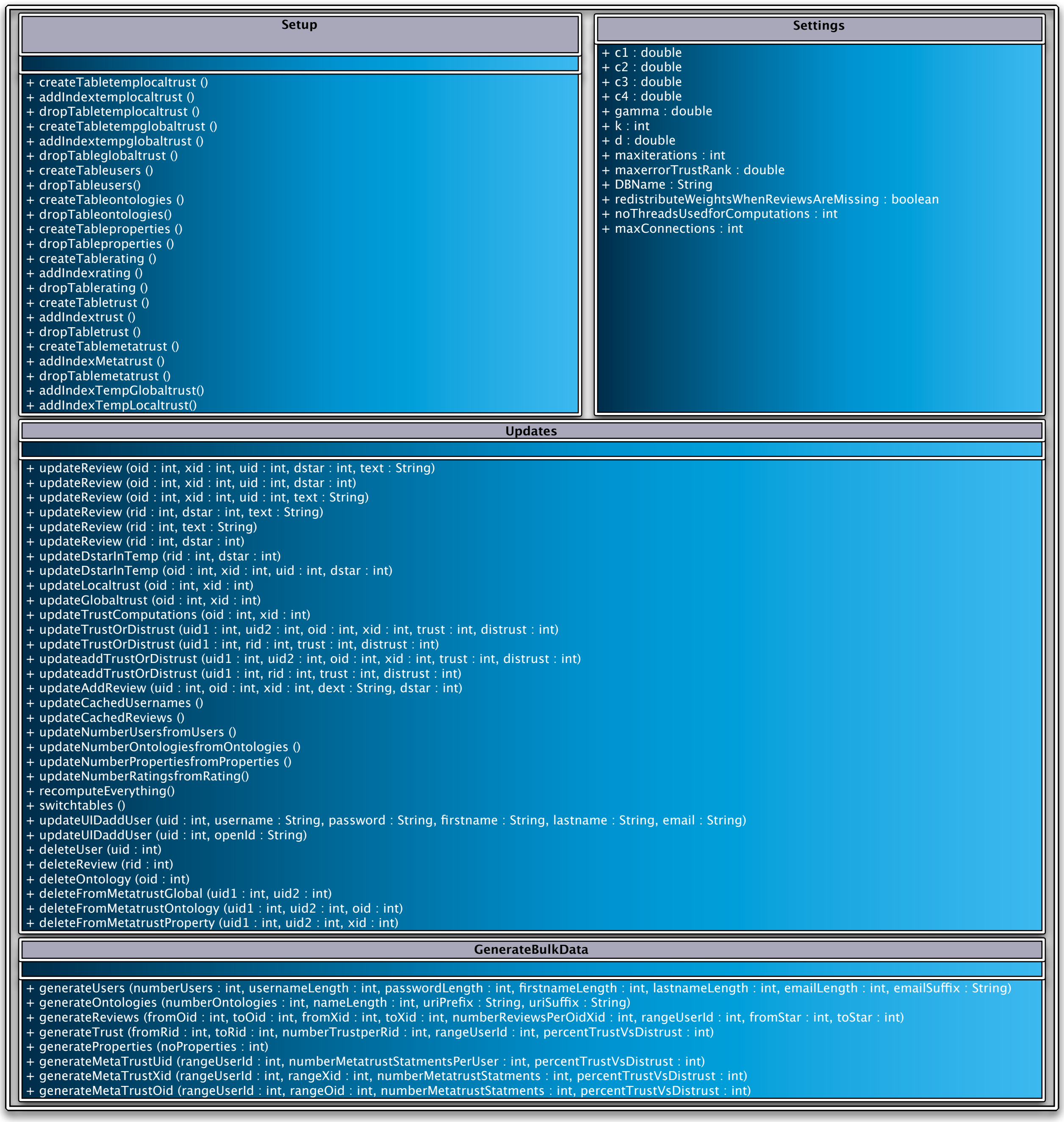

Setup
+ createTabletemplocaltrust ()
+ addIndextemplocaltrust ()
+ dropTabletemplocaltrust ()
+ createTabletempglobaltrust ()
+ addIndextempglobaltrust ()
+ dropTableglobaltrust ()
+ createTableusers ()
+ dropTableusers()
+ createTableontologies ()
+ dropTableontologies()
+ createTableproperties ()
+ dropTableproperties ()
+ createTablerating ()
+ addIndexrating ()
+ dropTablerating ()
+ createTabletrust ()
+ addIndextrust ()
+ dropTabletrust ()
+ createTablemetatrust ()
+ addIndexMetatrust ()
+ dropTablemetatrust ()
+ addIndexTempGlobaltrust()
+ addIndexTempLocaltrust()
Settings
+ c1 : double
+ c2 : double
+ c3 : double
+ c4 : double
+ gamma : double
+ k : int
+ d : double
+ maxiterations : int
+ maxerrorTrustRank : double
+ DBName : String
+ redistributeWeightsWhenReviewsAreMissing : boolean
+ noThreadsUsedforComputations : int
+ maxConnections : int
Updates
+ updateReview (oid : int, xid : int, uid : int, dstar : int, text : String)
+ updateReview (oid : int, xid : int, uid : int, dstar : int)
+ updateReview (oid : int, xid : int, uid : int, text : String)
+ updateReview (rid : int, dstar : int, text : String)
+ updateReview (rid : int, text : String)
+ updateReview (rid : int, dstar : int)
+ updateDstarInTemp (rid : int, dstar : int)
+ updateDstarInTemp (oid : int, xid : int, uid : int, dstar : int)
+ updateLocaltrust (oid : int, xid : int)
+ updateGlobaltrust (oid : int, xid : int)
+ updateTrustComputations (oid : int, xid : int)
+ updateTrustOrDistrust (uid1 : int, uid2 : int, oid : int, xid : int, trust : int, distrust : int)
+ updateTrustOrDistrust (uid1 : int, rid : int, trust : int, distrust : int)
+ updateaddTrustOrDistrust (uid1 : int, uid2 : int, oid : int, xid : int, trust : int, distrust : int)
+ updateaddTrustOrDistrust (uid1 : int, rid : int, trust : int, distrust : int)
+ updateAddReview (uid : int, oid : int, xid : int, dext : String, dstar : int)
+ updateCachedUsernames ()
+ updateCachedReviews ()
+ updateNumberUsersfromUsers ()
+ updateNumberOntologiesfromOntologies ()
+ updateNumberPropertiesfromProperties ()
+ updateNumberRatingsfromRating()
+ recomputeEverything()
+ switchtables ()
+ updateUIDaddUser (uid : int, username : String, password : String, firstname : String, lastname : String, email : String)
+ updateUIDaddUser (uid : int, openId : String)
+ deleteUser (uid : int)
+ deleteReview (rid : int)
+ deleteOntology (oid : int)
+ deleteFromMetatrustGlobal (uid1 : int, uid2 : int)
+ deleteFromMetatrustOntology (uid1 : int, uid2 : int, oid : int)
+ deleteFromMetatrustProperty (uid1 : int, uid2 : int, xid : int)
GenerateBulkData
+ generateUsers (numberUsers : int, usernameLength : int, passwordLength : int, firstnameLength : int, lastnameLength : int, emailLength : int, emailSuffix : String)
+ generateOntologies (numberOntologies : int, nameLength : int, uriPrefix : String, uriSuffix : String)
+ generateReviews (fromOid : int, toOid : int, fromXid : int, toXid : int, numberReviewsPerOidXid : int, rangeUserId : int, fromStar : int, toStar : int)
+ generateTrust (fromRid : int, toRid : int, numberTrustperRid : int, rangeUserId : int, percentTrustVsDistrust : int)
+ generateProperties (noProperties : int)
+ generateMetaTrustUid (rangeUserId : int, numberMetatrustStatmentsPerUser : int, percentTrustVsDistrust : int)
+ generateMetaTrustXid (rangeUserId : int, rangeXid : int, numberMetatrustStatments : int, percentTrustVsDistrust : int)
+ generateMetaTrustOid (rangeUserId : int, rangeOid : int, numberMetatrustStatments : int, percentTrustVsDistrust : int)

Figure 4.3: This graphic depicts a UML class diagram of the rest of the TS-ORS core components.

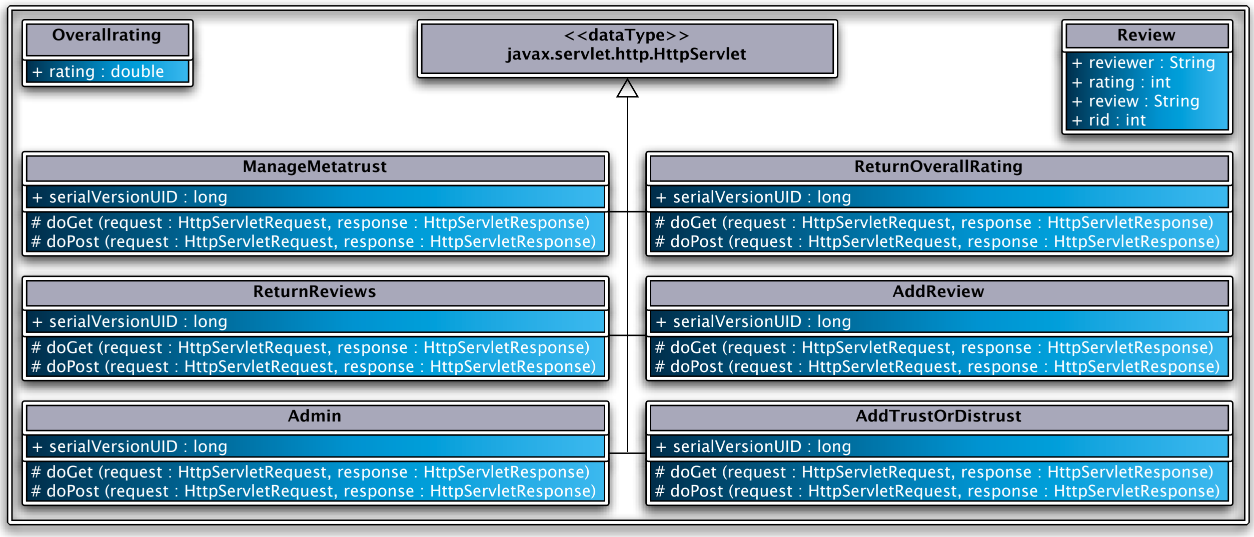

Overallrating
+ rating : double
<<dataType>>
javax.servlet.http.HttpServlet
Review
+ reviewer : String
+ rating : int
+ review : String
+ rid : int
ManageMetatrust
ReturnOverallRating
ReturnReviews
AddReview
Admin
AddTrustOrDistrust
+ serialVersionUID : long
# doGet (request : HttpServletRequest, response : HttpServletResponse)
# doPost (request : HttpServletRequest, response : HttpServletResponse)

Figure 4.4: This graphic depicts a UML class diagram of the servlets used for interaction with Watson.

4.2.3 Interaction with other Programs

As can be seen in Figure 4.4, the interaction with Watson is managed through Java servlets running on a Tomcat Server. There are 6 servlets exposing the functionality:

- **AddReview**: This servlet is called to add reviews to the database of the TS-ORS. In case the ontology is not known to the system, it is automatically added to the database.
- **AddTrustOrDistrust**: This servlet can add trust or distrust between two users for a specific ontology, property combination.
- **ManageMetatrust**: Using this servlet, metatrust information can be added to the database.
- **ReturnReviews**: Using this servlet, reviews can be retrieved for a certain ontology, property combination. If the user is logged in, they are returned in a personalized order, if not, they are returned based on global trust. Depending on content negotiation, the reviews are returned either as XML, JSON or HTML.
- **ReturnOverallRating**: Using this servlet, overall ratings are retrieved for the specified ontology and based on the parameters provided. If the user is logged in, they are returned in a personalized order, if not, they are returned based on global trust. Depending on content negotiation, the reviews are returned either as XML, JSON or HTML.
- **Admin**: This servlet can be used to access administrational functionality, such as triggering recomputation.

Furthermore, two classes called **Overallrating** and **Review** are used for storing the results before they are serialized as either XML or JSON.

4.3 Evaluation

Because the performance of the system is one of the most important aspects when serving a web application that provides results to the user, we have benchmarked the time to perform certain functionality. In order to get good estimates, we have seeded the tables with randomly generated data and could thus see how the number of users or trust statements or reviews affect the computation time.

4.3.1 Setup

We created a Java Class Evaluation that contains a method to run our evaluation setup. This way, people who have the code can run it and compare the results to the performance of their system. The method works by first filling the database with randomly generated data, based on predefined parameters. Parameters include the number of users, ontologies, reviews per ontology–property combination, trust statements on these reviews and metatrust statement as well as the percentage of trust vs. distrust. It is checked that no duplicate/contradictory data is generated during the process.

After the data is created, the computationally heavy steps are performed to compute the trustrank, distrustrank and localtrust for all ontology–property combinations. This step also has to be performed from time to time in running implementations to make sure the most current data is used for the ranking of reviews. The process involves multiple steps:

- Materialization of the temp database to allow faster access
- Setting Eigentrust (making sure every reviewer trusts his/her own review)
- Propagating metatrust and metadistrust statements (whilst ensuring that more specific statements are not overwritten by general statements)
- Computation of the trustrank, distrustrank and localtrust (including interpretation) for all ontology–property combinations in the database (since this task also relies mainly on CPU and not only on database, it is parallelized)
- Indexes are added to tempglobaltrust and templocaltrust
- The tables are renamed: temp becomes runtimetemp, tempglobaltrust becomes globaltrust, templocaltrust becomes localtrust. This way, the system can stay in operation when the trust information is recomputed.

Once this data is computed, the runtime tests are performed. We measure how long it takes to retrieve an overall rating for all ontologies, both using global and local trust. The tasks are performed 300 times based on a varying number of reviews (using only top, top 3 and all reviews). We measure the duration of each execution and report the minimum, maximum and average duration. It is important to see how database caches can be exploited to minimize the latency during runtime. The maximum time usually is caused by the results not being cached, while the rest of the runs the results of the database query should still be cached. We can exploit that feature by querying all ontologies in the system once after recomputation, so that many results will be cached and runtime database access can be minimized. We also measure how long it takes to retrieve all reviews for all ontology-property combinations based on both global trust ranking and local trust ranking.

4.3.2 Results

We have performed the benchmark on two systems: A Dualcore MacBook Pro running Mac OS X 10.5.6 and a QuadCore PC running Vista 64bit.

In order to find out how a different number of reviews for each item and trust statements on these reviews would influence the computation time, we did not only vary the number of users (100, 250, 500), but also the amount of reviews and trust statements available, resulting in 9 different testruns. During the generation of the test data, for each different user group size, we had one setting which had 10% of the users review each ontology–property combination and 10% of the users then trusting each of these reviews, one with a 50%–50% distribution and a worst case scenario (100%–100%). Worst case means that each user reviews every ontology–property combination, and also every user then votes on the usefulness of these reviews. In a realistic setting, the distribution is likely less than 10%–10%. For the metadatatrust generation, we have fixed the percentage of global, ontology- and property-specific metatrust statements to 20% per user, i.e., each user metatrusts 20% of the other users. The test data was regenerated between all runs, to prevent caching effects in between runs. As for the number of properties, they were fixed to 5. Furthermore we limited the number of ontologies to 12, since this was the smallest number allowing to distribute the computations evenly among 4 threads. Furthermore, the complexity of the computation does not increase by having more ontologies, this is just a linear factor (it will take 10 times longer to perform the computations for 120 instead of 12 ontologies). This is because all the computations have to be performed for each ontology–property combinations.

MacBook Pro The computation was performed on a MacBook Pro with 2.16 GHz Intel Core 2 Duo Processor and 3GB 667 MHz DDR2 SDRAM running Mac OS X 10.5.6. Harddrive is ST9320421ASG (320GB 7200 RPM, 16MB Cache, avg seek time 11 ms). Eclipse Version is 3.3.2 for Mac. MySQL versions are 5.0.45 for the server and 5.1.5 for the J-Connector.

The results are shown in the figures 4.5, 4.6, 4.7, 4.8 and 4.9. The percentages in the labels refer to the percentage of reviews, trust- and metatrust statements generated as test data (see explanation above). For example 10%10%20% means that 10% of all users have reviewed each ontology, 10% of all users have rated each review, and each user expresses metatrust towards 20% of the other users. We believe that this setup allows to draw some conclusions about the scalability of the system with regard to number of users and sparsity of data. In the following section we will analyze the results seen in the figures presented here.

QuadCore Vista 64bit The computation was performed on a 2.40 GHz Intel Core 2 Quad Q6600 MacBook Pro with 8GB 800 MHz DDR2 SDRAM running Microsoft Vista Business 64bit. The harddisk is a Samsung HD103UJ (1TB, 7200 RPM, 32MB Cache, avg seek time 8.9). Eclipse Version is 3.4.1 for Windows. MySQL versions are 5.1.31 for the server and 5.1.7 for the J-Connector.

4.3.3 Analysis

We will now analyze the results of the benchmark and try to explain the system behavior. We will start with Figures 4.5 and 4.10, which depict the computation of trust and the propagation of metatrust. In order to understand which steps are performed in these two events, we will now explain them in more detail.

Metatrust Propagation The general idea behind the concept of metatrust is saving the user the time and effort to find all reviews by a particular user and state that he or she trusts them. So each metatrust statement can be seen as a number of particular trust statements towards the reviews covered by the metatrust statement. So if a metatrust statement is made for a particular user and ontology combination, the metatrust statement has to be transferred into several trust statements on all properties of the ontology this user reviewed. Another important fact is that more special trust or metatrust statements cannot be overwritten, and that distrust is more important than trust. This is important to allow constructs like trusting another user globally except for one ontology or property. In order to take all these limitations into account, the propagation of metatrust is performed as follows: First all metadistrust statements covering properties of ontologies are retrieved from the metatrust table and propagated. This is done by retrieving a list of reviews that the user that is trusted has written for this property, and removing the reviews that are already trusted or distrusted by the user issuing the trust statement. Then for all the remaining reviews, a distrust statement is added. Afterwards, the same is done for metatrust statements covering the properties, metadistrust and -trust statements covering ontologies, and last the global metadistrust and -trust statements. As expected, the time taken to perform this operation increases with the number of users and the amount of trust statements already in the database. The main limiting factor here are the database lookups needed to check each time, whether a more specific trust statement already exists, before propagating the metatrust.

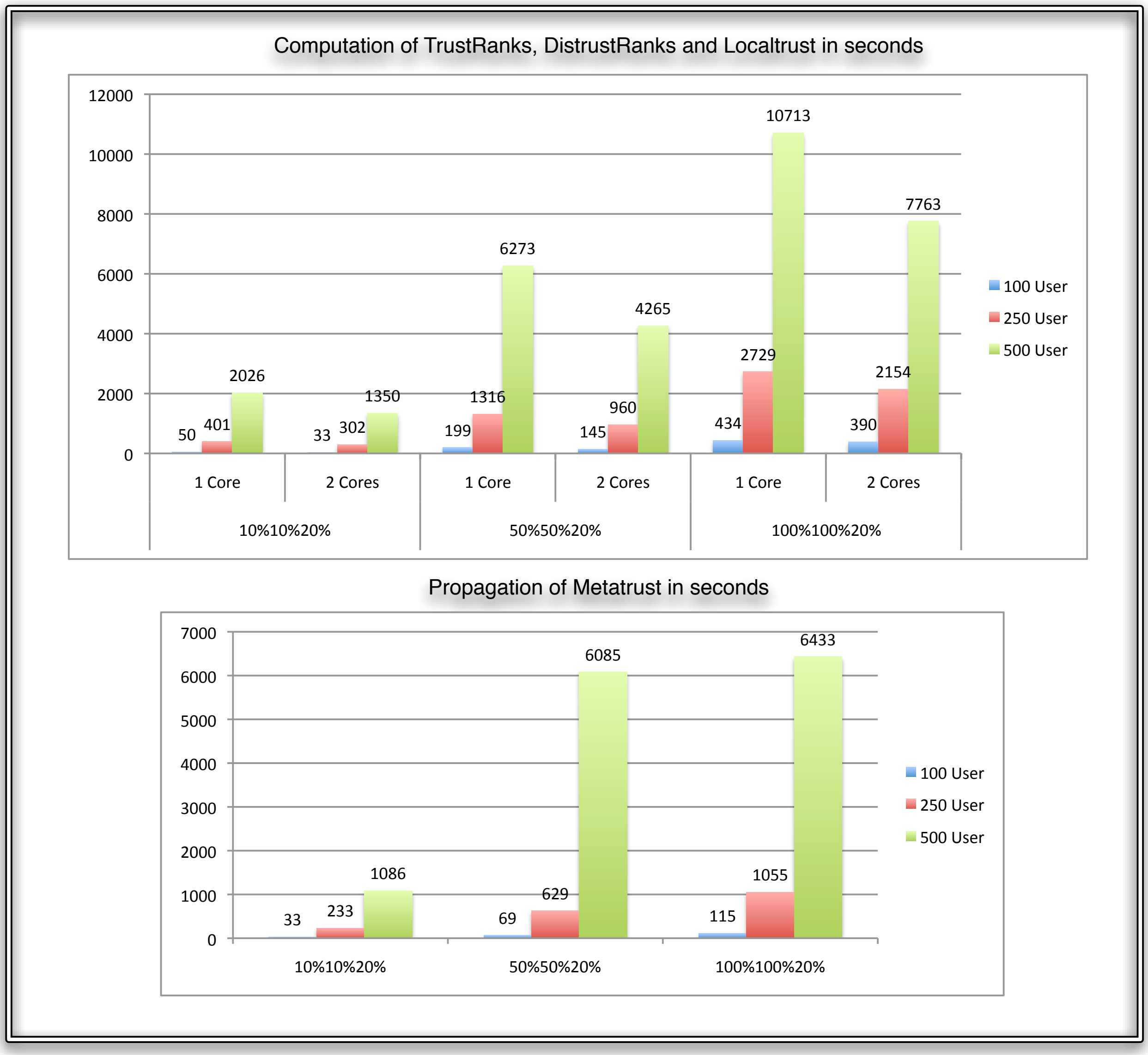


Figure 4.5: This graphic depicts the results of the benchmark of the computation and metatrust propagation. Data is presented as time in seconds. (Run on MacBook Pro)

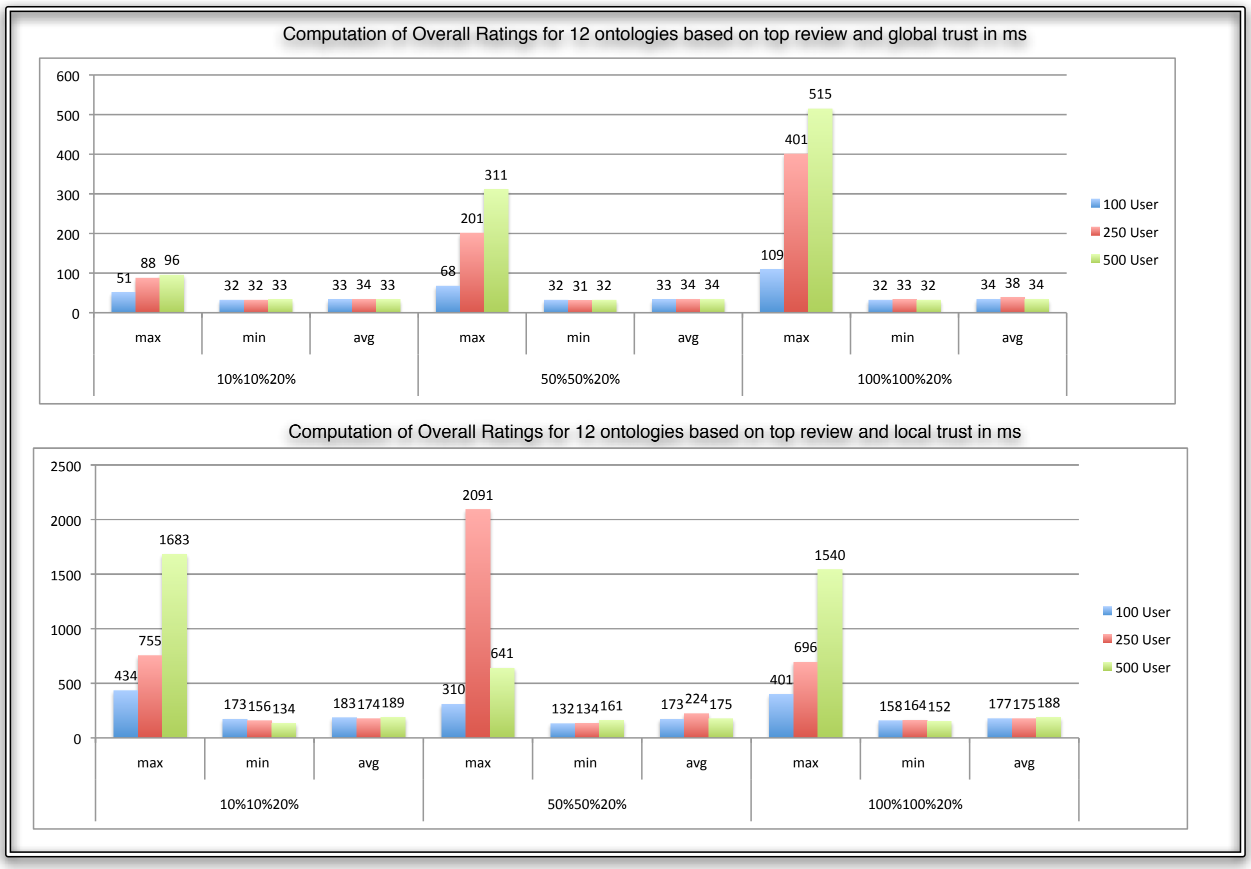


Figure 4.6: This graphic presents the times taken (min, max and avg) for returning the overall rating for all 12 ontologies in ms once for global and once for local trust. 300 runs were performed and only the top review was considered for the computation. (Run on MacBook Pro)

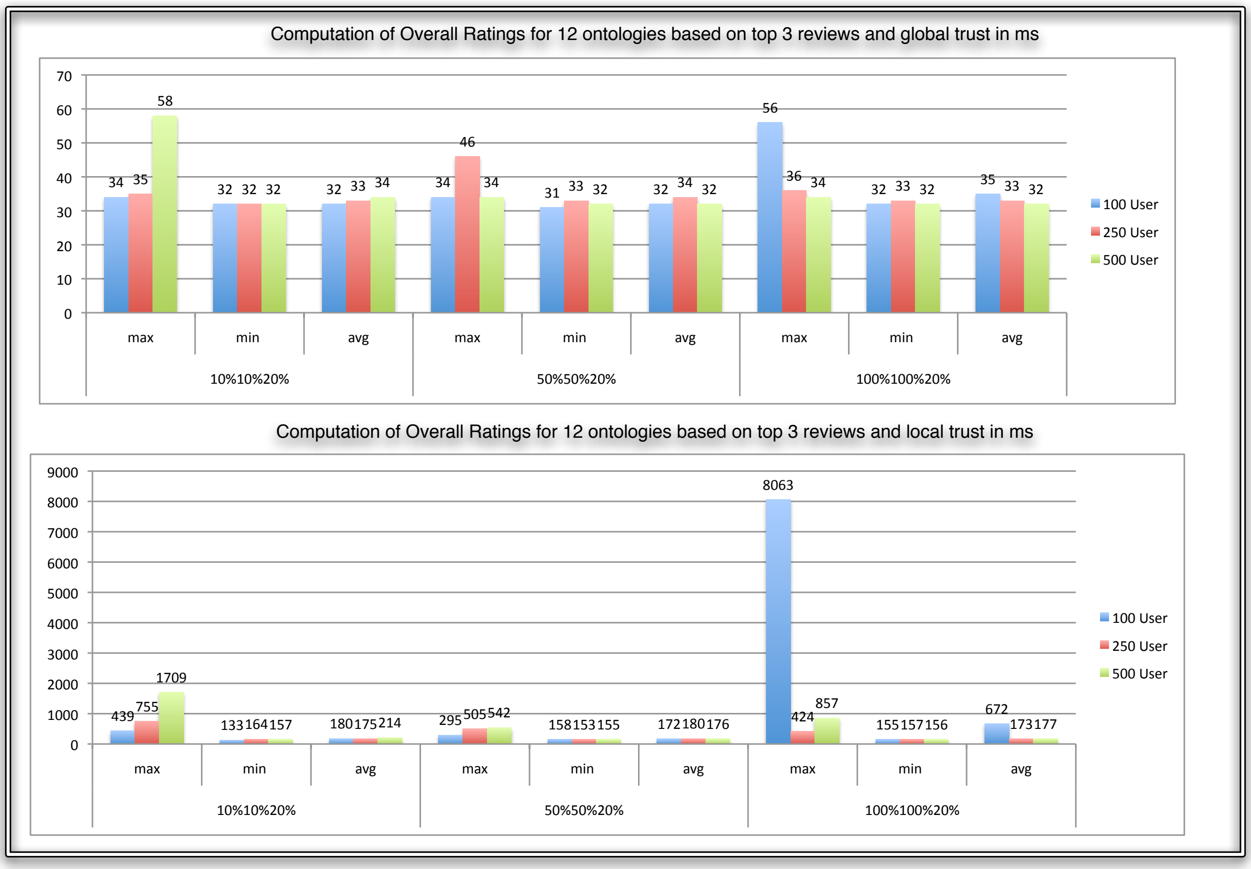


Figure 4.7: This graphic presents the times taken (min, max and avg) for returning the overall rating for all 12 ontologies in ms once for global and once for local trust. 300 runs were performed and only the top 3 reviews was considered for the computation. (Run on MacBook Pro)

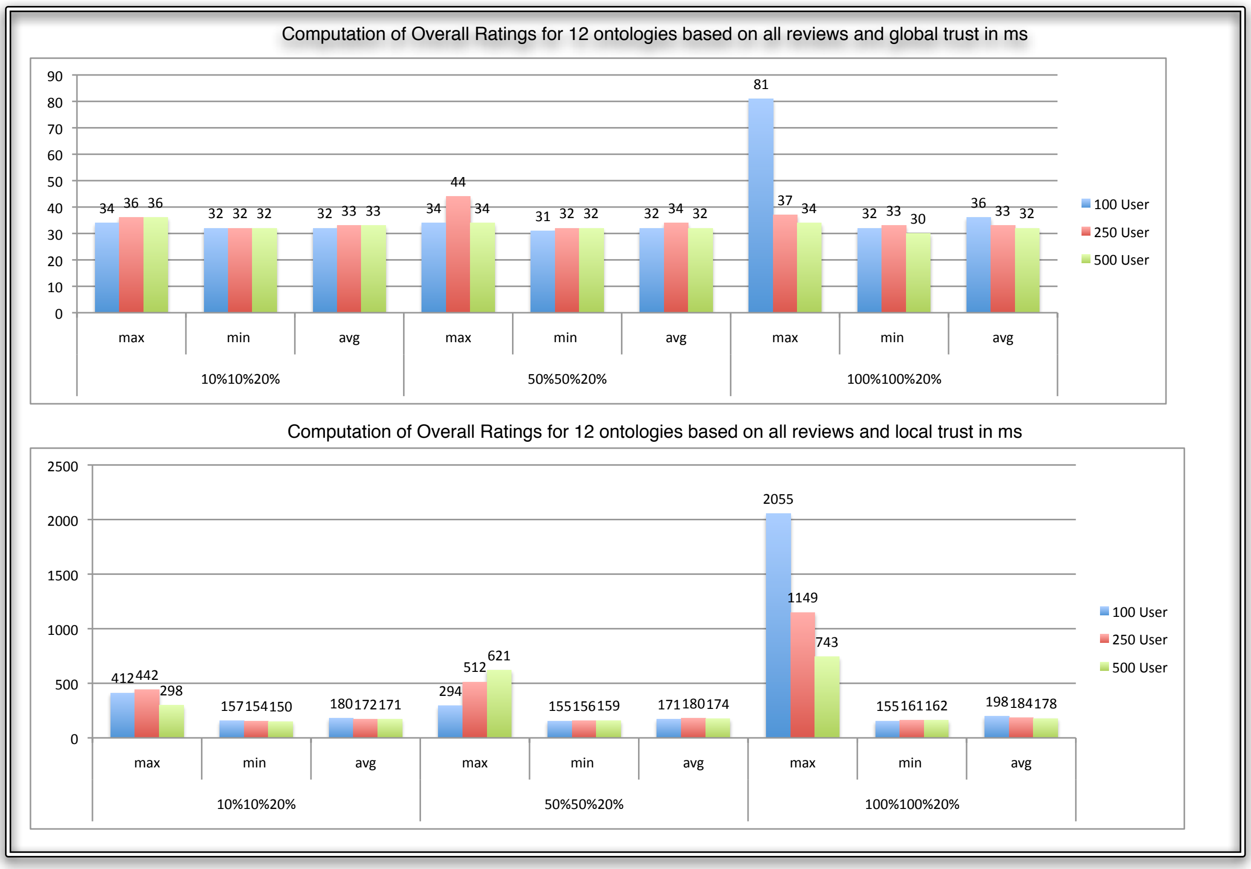


Figure 4.8: This graphic presents the times taken (min, max and avg) for returning the overall rating for all 12 ontologies in ms once for global and once for local trust. 300 runs were performed and all reviews was considered for the computation. (Run on MacBook Pro)

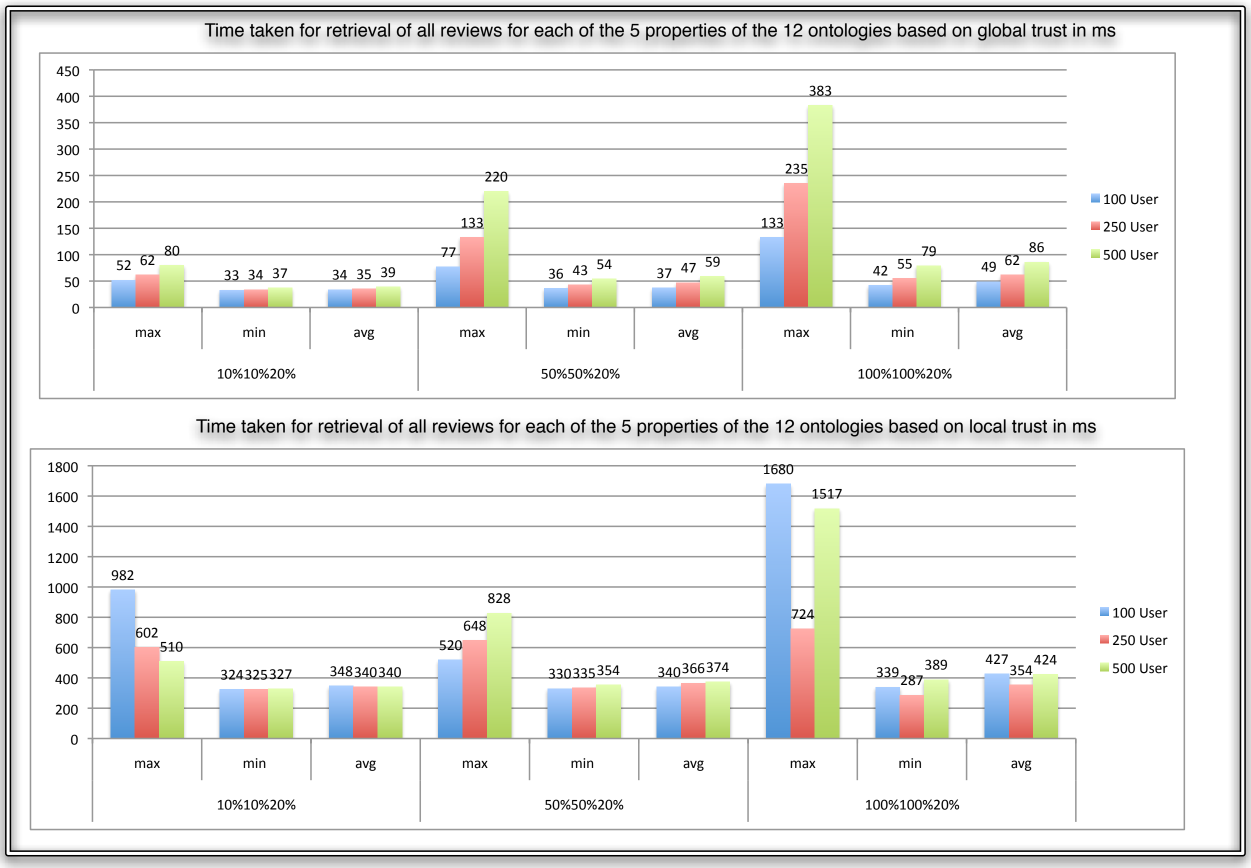


Figure 4.9: This graphic presents the times taken (min, max and avg) for returning all reviews for all 5 properties of all 12 ontologies in ms once based on global and once based on local trust. 300 runs were performed. (Run on MacBook Pro)

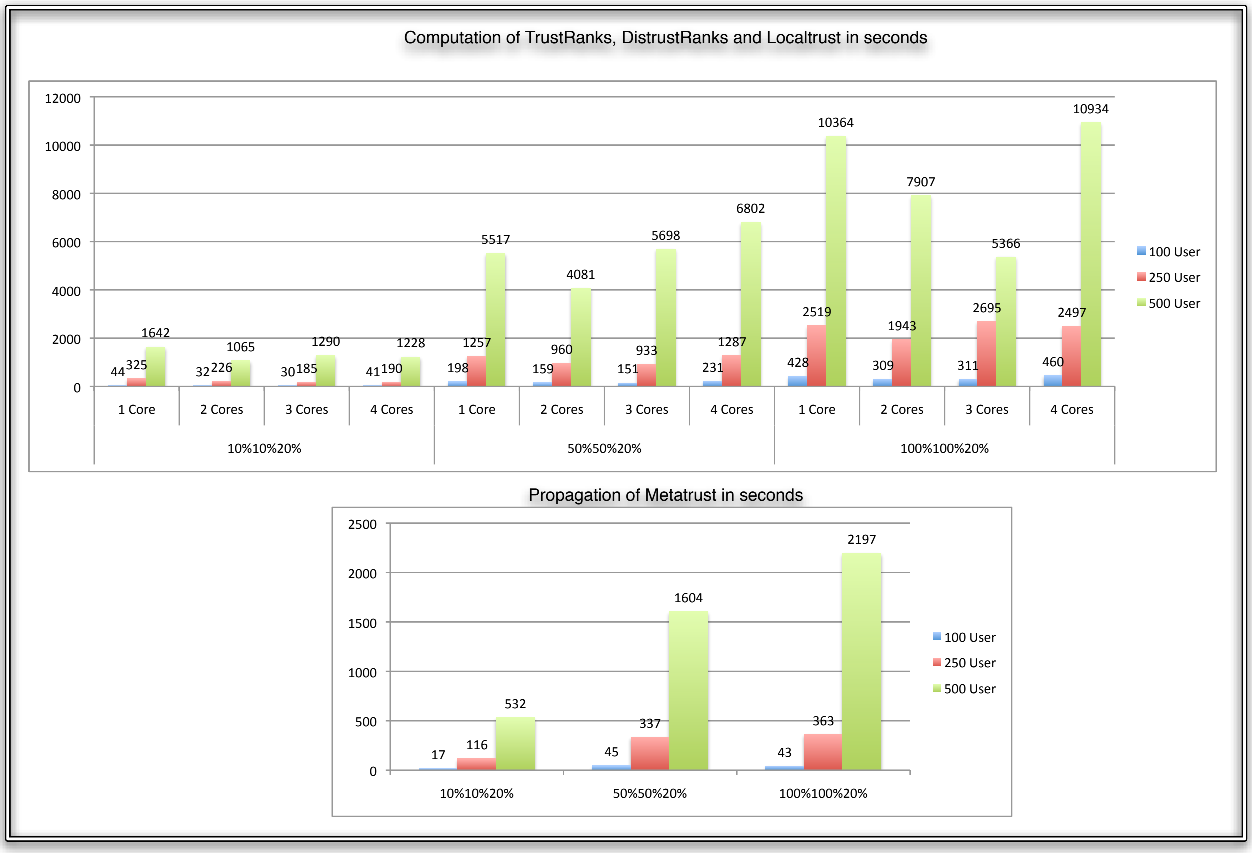


Figure 4.10: This graphic depicts the results of the benchmark of the computation and metatrust propagation. Data is presented as time in seconds. (Run on Intel Core 2 Quad)

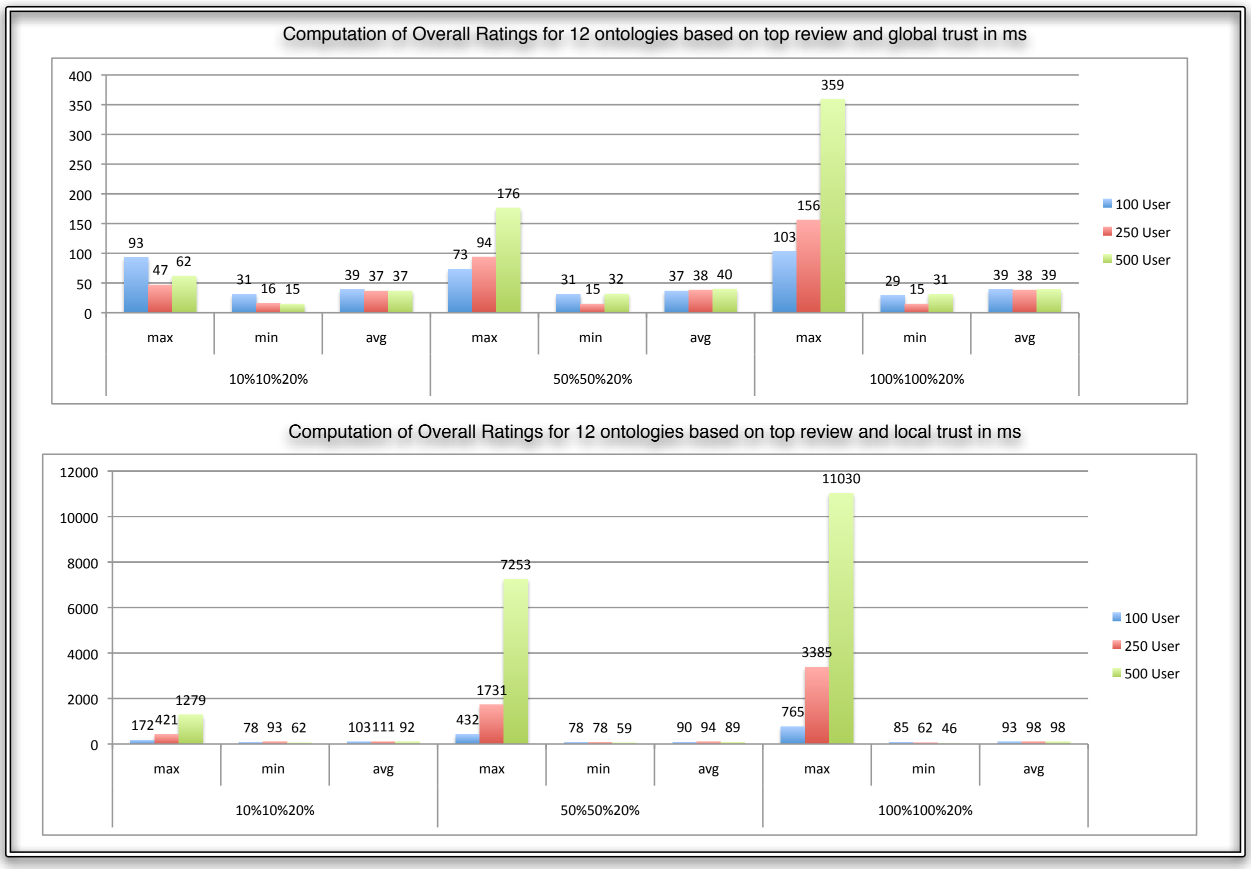


Figure 4.11: This graphic presents the times taken (min, max and avg) for returning the overall rating for all 12 ontologies in ms once for global and once for local trust. 300 runs were performed and only the top review was considered for the computation. (Run on Intel Core 2 Quad)

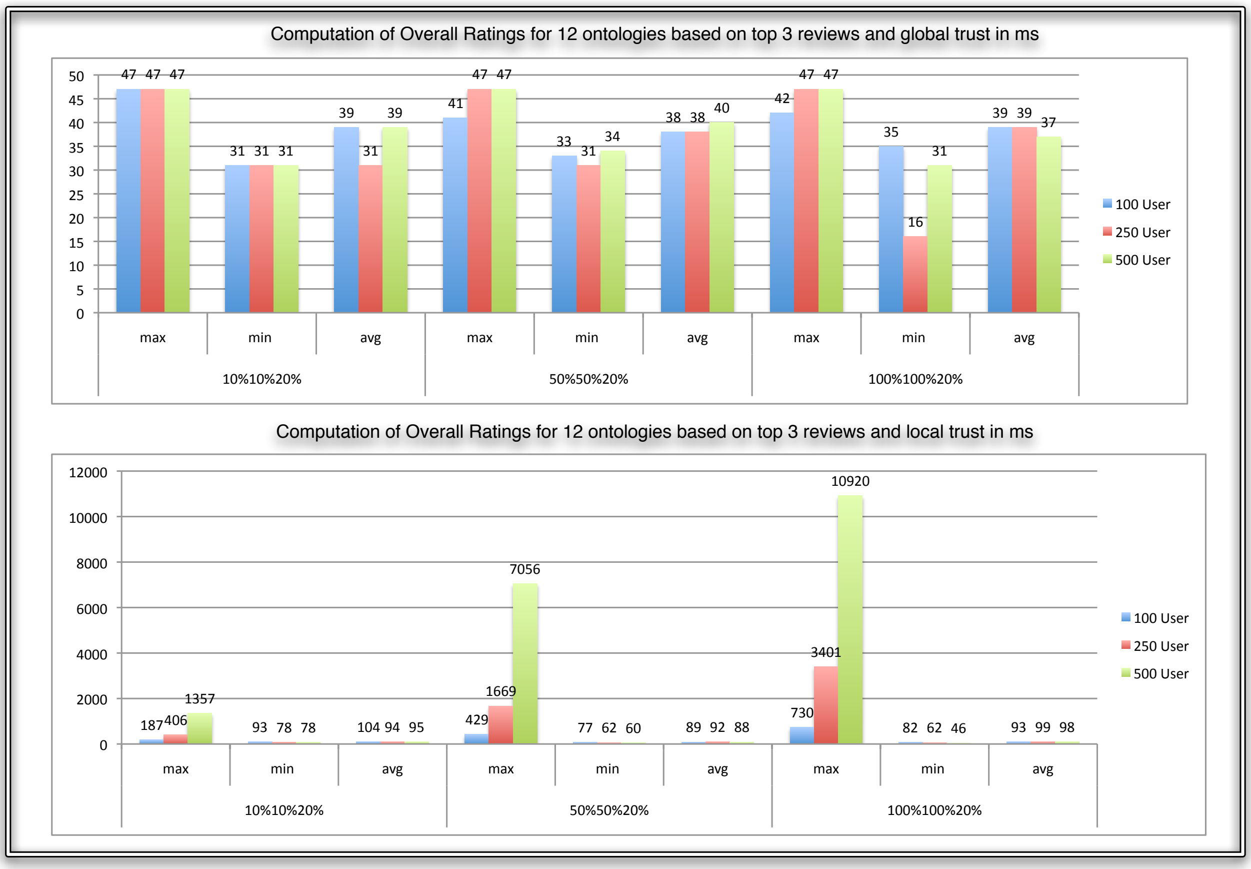


Figure 4.12: This graphic presents the times taken (min, max and avg) for returning the overall rating for all 12 ontologies in ms once for global and once for local trust. 300 runs were performed and only the top 3 reviews was considered for the computation. (Run on Intel Core 2 Quad)

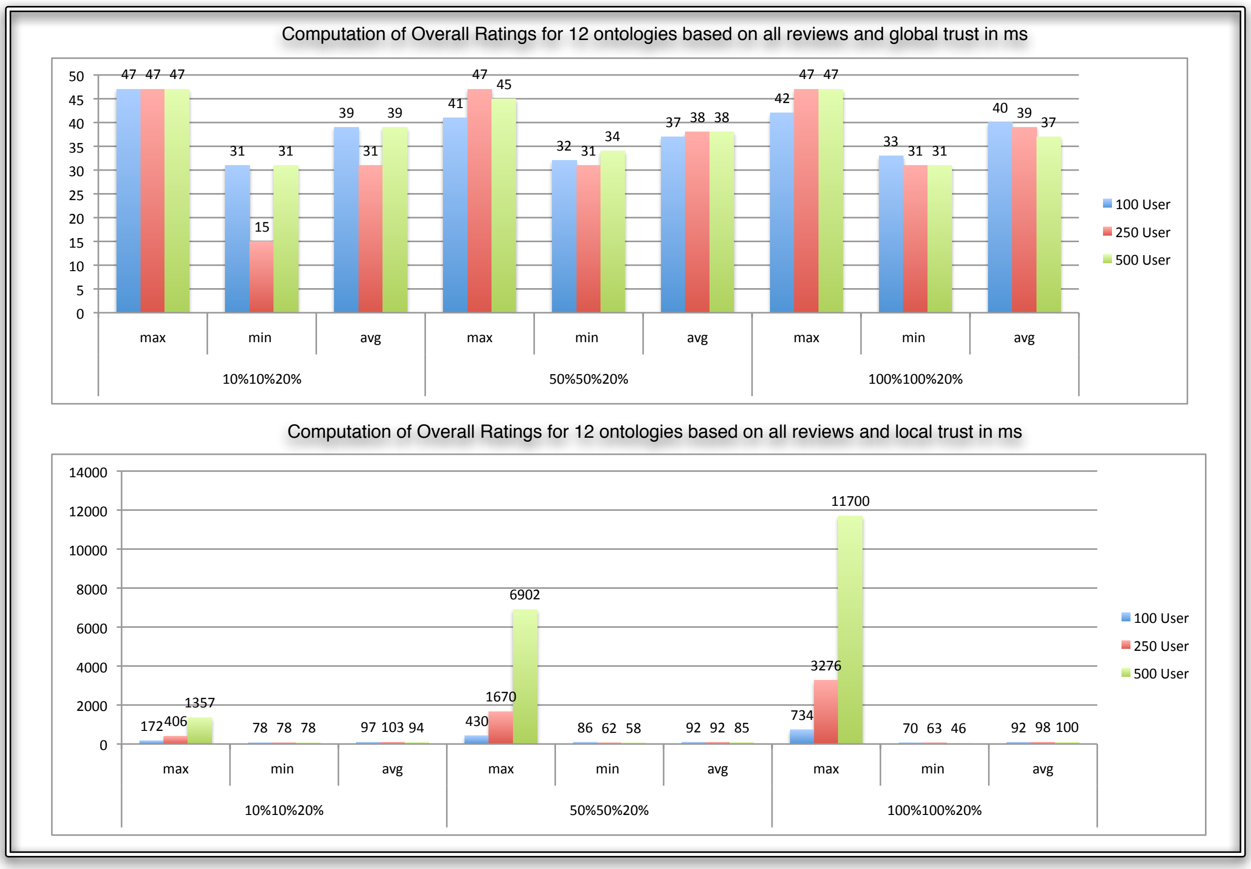


Figure 4.13: This graphic presents the times taken (min, max and avg) for returning the overall rating for all 12 ontologies in ms once for global and once for local trust. 300 runs were performed and all reviews was considered for the computation. (Run on Intel Core 2 Quad)

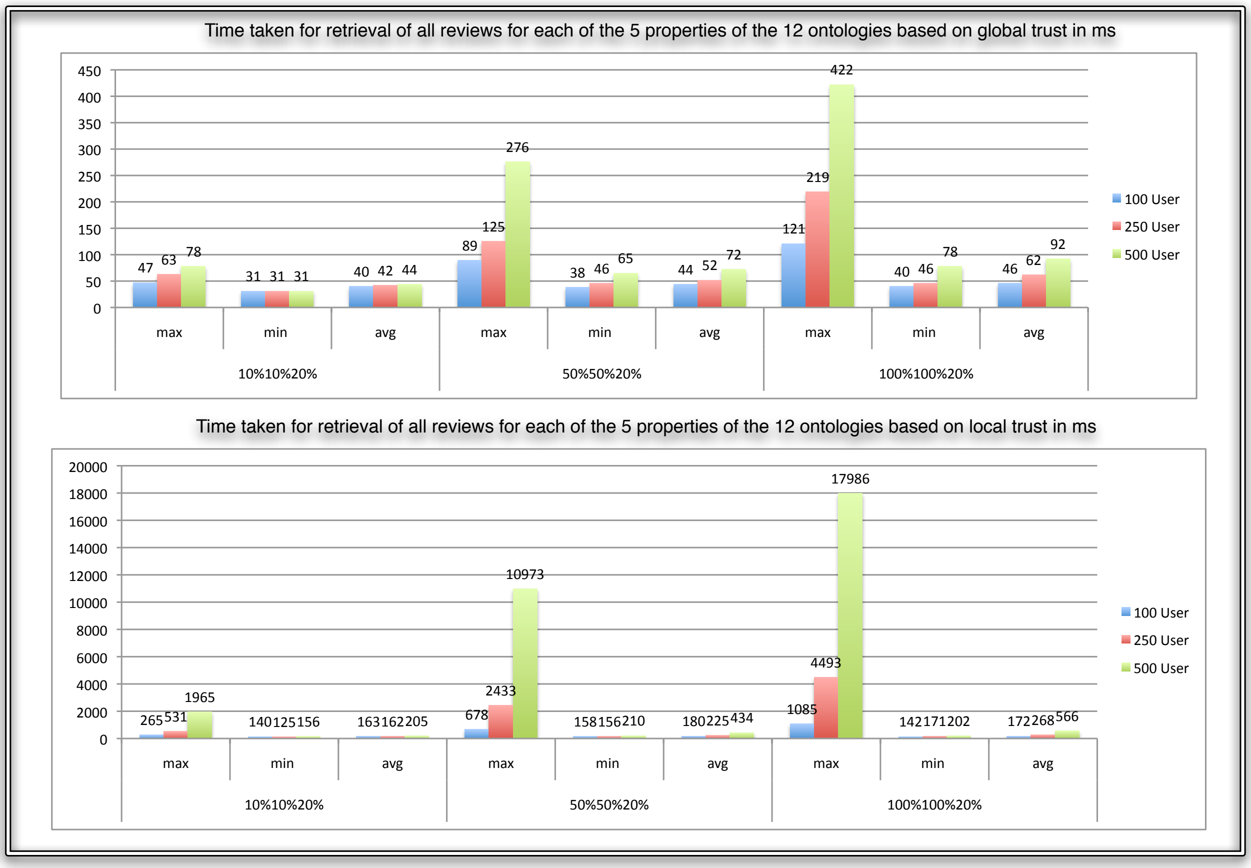


Figure 4.14: This graphic presents the times taken (min, max and avg) for returning all reviews for all 5 properties of all 12 ontologies in ms once based on global and once based on local trust. 300 runs were performed. (Run on Intel Core 2 Quad)

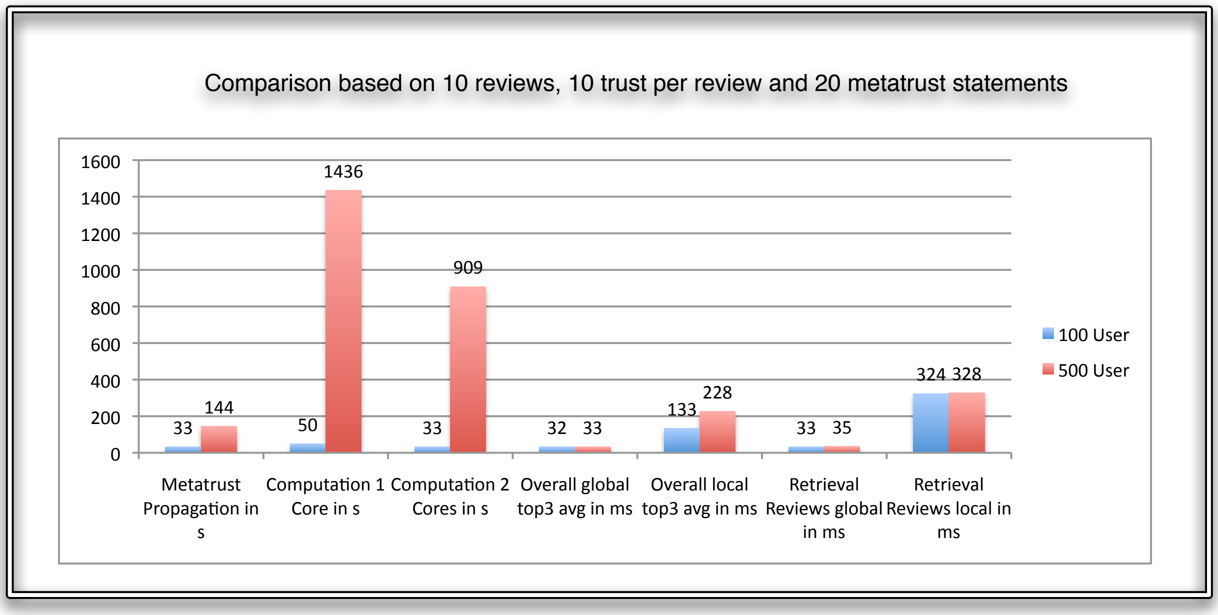


Figure 4.15: This graphic presents a comparison between the time taken for different actions based on the same review and trust statements but increased number of users. (Run on MacBook Pro)

The results indicate what was expected, namely that the duration increases if more reviews are in the system, and also if the number of users increases. Since this method almost solely relies on database operations, it can profit from the faster disk and faster disk access of the desktop harddisk. The times are at least twice as fast on the better machine. In our example, the duration of execution is growing a little bit more than squared in the relation to the number of users. However, this is mainly due to the bigger number of reviews and trust statements that we set to a fixed portion of the users. So for the 10%10%20% that means that in the case of 500 users, we have 5 times the amount of reviews, trust- and metatruststatements. If we only increase the number of users, but not the number of reviews, the computation only lasts slightly longer. For example, the propagation of metatrust for the setting 500 users with a 2%, 2%, 4% setting (which is equivalent to the 100 User 10%10%20% setting) takes 144 seconds instead of 33 for the case of 100 Users (which is less than linear growth with regard to the increase in number of users). So in case the number of reviews is small in comparison to the userbase, an increase in users does not affect the computational time too much. The results for this comparison can be seen in figure 4.15.

Trust Computation For the technical details of the computation, please see [SAd+07]. The benchmark results can be found in figure 4.5 and 4.10. The computations are the backbone of the whole TS-ORS system. After the metatrust is propagated, all trust statements are the most special form, i.e. on reviews of a specific ontology–property combination. In the trust computation phase, for each of the ontology–property combinations, the trustrank, distrustrank and localtrust are computed. Computing the trustrank is basically computing PageRank [PBMW98]. In order not to spend unnecessary time during the iterations needed to converge to a solution, we have implemented 2 stop criteria, which can be specified in the settings class. Either the overall change in trustrank (cumulated difference of ranks between two computations) is below a certain threshold, or a maximum number of iteration is reached.

The distrustrank can be computed in one iteration after the trustranks are known.

The localtrust computation relies heavily on matrix multiplication. Also the results have to be interpreted, which is based on known trust and distrust statements and a comparison of unknown values to neighbors.

To see how multiple cores could share the computational effort, we have distributed the computation among up to 4 cores, and checked how time was affected. While the threads each had their own data in memory, they all accessed the same database (using a connection pool) to retrieve the trust and distrust information needed for the computation.

The results indicate that using 2 threads does indeed lead to faster computation. However, using 3 or 4 threads even on the 4 core machine mainly takes longer than just 2. It seems that the database access is the limiting factor here. If 4 threads try to read and write information from and to the database, they have to wait until the database can handle the request. So in this case having distributed databases that are able to handle the database interaction in a timely fashion would increase the performance more than having more threads on more CPUs.

The results seen in the figures are for computing all 60 ontology–property combinations. So you have to divide the results by 60 to know how long one trust computation cycle lasts. For updates during runtime (a single trust statement was added), it is possible to recompute the trust on the fly, since the time it takes for a realistic setting (few reviews and trust statements) should be less than 10 seconds (even under a second for the 100 user case with sparse data).

Overall Computation In order to see how the different parameters influence the time for retrieving the overall rating for an ontology, we have based the computation on only the top review, the top 3 reviews and all reviews (just for worst-case considerations). For each of these settings, we base the computation on both local trust and global trust. Furthermore, we run each computation 300 times, to get more accurate average results. We measure the execution time for each of the 300 runs, and present the maximum time as well as average and minimum time needed for providing the result. It is intended to make use of the databases caching techniques, since they can also be exploited in real systems (how will be explained later). For the local trust based computation, we changed the user for which the results were computed in between the 3 different settings (top-1, top-3, all) so that results would have to be re-cached). The results can be found in Figures 4.6, 4.7, 4.8, 4.11, 4.12 and4.13.

The overall computation is one of the most important features of the TS-ORS and it is performed constantly at runtime. So here a quick response time is far more important than for the larger computations which are performed offline and only at dedicated time-points.

The results indicate that while the maximal time in case of a cache miss or for other system-specific reasons can be in the order of seconds (bear in mind that this is for all 12 ontologies), the more significant average is relatively independent of the amount of users or the number of reviews in the system. The only difference is that retrieving the result based on global trust is roughly 5-10 times faster than retrieving the result based on local trust. Is is due to the fact that for local trust more database queries have been made, if the top reviews cannot be found within the first query. The way the top reviews are found differs between local and global trust.

For global trust, retrieving the top reviews is simply one query to the globaltrust database with a weighing parameter alpha. Then an ordered result is retrieved that can be used for computation. In case of local trust, first the top locally trusted reviews are retrieved, then all possible remaining globaltrust results, and lastly the locally distrusted reviews.

Now considering a realistic scenario (e.g. the 10%10%20% Top 3 reviews), based on global trust the score for one ontology can be retrieved in roughly 3 ms and for local trust in roughly 8 ms. These values are more or less independent of the user size. Given that all 3 reviews are found in the first part of the local trust query, the results for local trust can be even less than 8 ms.

Review Retrieval When a user wants to browse reviews for ontology–property combinations, it is important to retrieve them in a personalized order. So when a user is logged in, the reviews are retrieved in an order based on local trust of this user, otherwise they are ordered according to globaltrust. We have benchmarked retrieving all of the reviews in aforementioned personalized order for all ontology-property combinations. In Figures 4.8 and 4.9, you can see the results both based on localtrust and globaltrust. Again, we have run each task 300 times to compute accurate results.

This task will also often be requested at runtime, when a user is browsing different ontologies. Since all the reviews have to be ordered, results are better when there are not too many reviews. For a realistic scenario (e.g. the 10%10%20%), the time to retrieve the ordered reviews is about 0.7 and 1.5 ms for each ontology–properties based on global trust (depending on number of users and thus reviews). For retrieving the results based on local trust, between 2.7 ms and 7 ms are needed on the fast machine.

4.3.4 Results Learnt from Benchmark

Since many operations rely on a fast database, having good harddrives and enough caching enabled for the database is key to obtaining a good performance. In order to minimize query time at runtime, after each computation of trust, all ontology–property combinations could be queried once, to have the results stored in cache. This would lead to fast runtime response times.

It also seems sensible to use at least a dual-core machine with 2 threads for the trust computation, since the results are much faster than those obtained for a one-core solution. Depending on how important it is to take the latest user data into account, the frequency of overall recomputation can be increased or lowered. Amazon.com, for example, need 24 hours to take a trust statement into account. In case a really fast recomputation is needed, the computation can be distributed among different machines, each containing a database filled with only the necessary information. Since the computation of trust is independent from all ontology–property combinations, in the most extreme case, one could use one machine per combination and later merge the results.

The runtime performance looks promising, since most of the time values will only be asked for preselected ontologies that were in the resultset of a user-query. In case this resultset is limited to e.g. 50 ontologies, the overall ratings could be generated in less than 500 ms. Also retrieving the reviews is done very quickly. So the overall performance is satisfying. Nevertheless, we will see how to optimize the code even further.

4.4 Future Work

The TS-ORS will be evaluated within a reuse experiment performed in NeOn's Workpackage 5. While it is too early to report results here, they will be available in an upcoming deliverable. Furthermore, there are ongoing discussion to integrate the TS-ORS into the Ontology Designpatterns website[3], as part of the design pattern review process.

[3] `http://ontologydesignpatterns.org`

Chapter 5

Trust Propagation On Conceptual Structures

Trust frameworks like the one presented in the previous chapter consider trust as a relation between agents interacting with ontologies. However, in the context of Semantic Web applications, which consume knowledge accessed through gateways such as Watson, the question "what does it mean to trust an ontology?" has not yet been properly addressed. In other words, while approaches such as ReGreT [SSG07] provide a generic framework for reasoning with trust relations in a community of agents, they do not take into account the specific aspects characterizing ontologies, where formal relations exist between entities, which can be taken into account when assigning trust to them.

Here we use the trust assigned to individual statements by particular users and community of users, and propagate them as evaluation of entities. The basic usage scenario is the one of the Watson Plugin. On the one hand, the assignment of trust to statements can be realized in this context in a way that does not interfere the users' interaction with the tool. Indeed, we consider that whenever a user integrates a statement in her own ontology using the Watson Plugin she implicitly indicate that she trusts this particular statement. On the other hand, relying on a common trust engine, this trust information can be collected from all the different users of the Watson plugin, propagated as evaluations of the attached entities, and used to provide ranking and quality based information within the tool, facilitating in this way the task of selecting statements for reuse.

5.1 A trust engine for ontology entities

In [LMD$^+$09], we described a trust propagation model, to derive trust values for entities in an ontology based on trust values provided by users on individual statements. This model is based on the formal semantics of the ontological elements manipulated, deriving for example the trust value for a class based on the trust value for its (direct or indirect) instances. We implemented this formal model in a trust engine, deployed as a service on top of the Watson gateway.

More precisely, this trust engine provides functions to post evaluations on individual statements and to get trust values for statements and entities.[1] In the following we use three of the provided functions:

- The input function

$$evaluate(onto, s, p, o, user, value)$$

  where $onto$ is the URI of the ontology containing the statement to evaluate, s, p and o are the subject, predicate and object of the statement respectively, $user$ is the ID of the user providing the evaluation and $value \in [0..1]$ is the trust value associated with the statement. This function simply assigns the trust value $value$ to the statement for the user identified by $user$.

[1] Implemented in an HTTP/REST API.

- The output function

$$getEvaluation(onto, e, user) \rightarrow value : real,$$

which returns the trust value for a particular entity e, calculated by propagating the trust values the user $user$ has provided for statements formally related to e.

- The output function

$$getGlobalEvaluation(onto, e) \rightarrow value : real,$$

which returns the trust value for a particular entity e, calculated by propagating the trust values provided by any user on statements formally related to e.

In this model, the trust values, both given as input and computed by propagation, are real values between 0 and 1, where 0 means complete distrust, 1 means complete trust and 0.5 is a neutral value indicating that the user has no particular opinion on a statement or an entity. Although this can be parametrized, 0.5 is generally used as a default value.

5.2 Assessing statements in the Watson Plugin

The goal of the integration of our Trust Engine to be used within the Watson Plugin is to exploit interaction between the user and the system to provide positive evaluations for the statements that are being reused from external ontologies. By using a shared Trust engine, we can then collect these evaluations from the entire community of users, each user contributing in building a collective base of quality information about ontologies' entities and statements.

The overall architecture resulting from this integration is therefore relatively straightforward (see Figure 5.1). Each instance of the Watson plugin for the NeOn toolkit is associated a user ID and is connected to the Watson Trust Engine. As mentioned above, we want to exploit the interaction between the user and the system to collect trust statements in such a way that it does not obstruct the normal use of the Watson Plugin, which main goal remains to facilitate reuse in building ontologies. This is achieved by interpreting the action of "integrating an external statement" in the base ontology as a positive trust indication the user emits concerning this particular statement. Therefore, such an evaluation is automatically sent to the Trust Engine, as a side effect of the user reusing this statement for his/her own ontology.

5.3 Trust based ranking in the Watson Plugin

There are many ways in which the collected quality information can be used in various applications. In the Watson plugin, the evaluations the user has provided, as well as the ones from other users, are used to rank the entities from which statements to be reused are proposed. It provides quality information on these entities in the form of trust values, either aggregated from the community of users, or personal to the current user.

To achieve this, each ontological entity retrieved using the Watson plugin is assigned a score, which is based partly on the user's previous evaluations and partly on the global evaluation, coming from the entire community of users:

$$score(onto, e, user) = W_u \times getEvaluation(onto, e, user) + W_g \times getGlobalEvaluation(onto, e)$$

We chose to use a higher value for W_u (0.7) compared to W_g (0.3) so that, for entities for which no evaluation has been provided by the user (i.e., $getEvaluation(onto, e, user) = 0.5$), a ranking is obtained from the

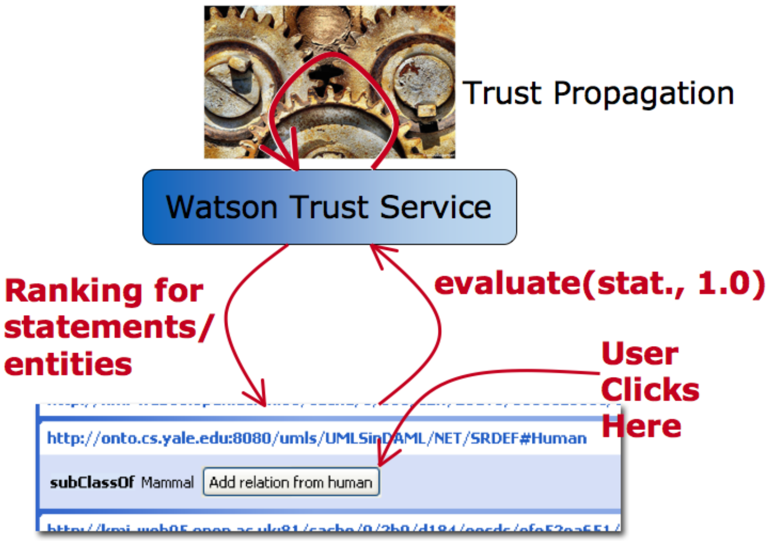


Figure 5.1: Overview of the integration of trust on statements and entities in the Watson Plugin.

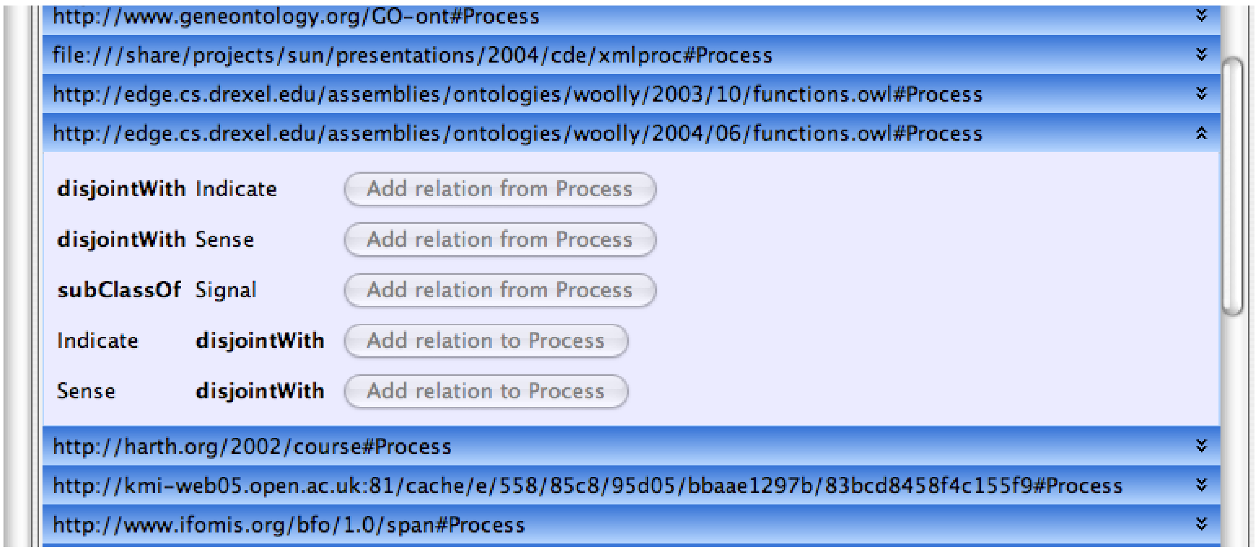
global ‘opinion’ from the community, while if the user has provided trust values related to the entity, his own ‘opinion’ should prevail.

Figures 5.2, 5.3 and 5.4 summarize a concrete example of how the integration of trust information is realized within the Watson plugin. In the first step (Figure 5.2), a user is looking for statements concerning the class $Process$. One of the retrieved entity (`http://edge.cs.drexel.edu/assemblies/ontologies/woolly/2004/06/functions.owl#Process`) is of particular interest for the user, which reuse three of the associated statements. Reusing these statements leads to the system sending positive trust values for each of them to the trust engine (using the $evaluate$ function). Having done that, whenever the user will search again for the $Process$ class, the `http://edge.cs.drexel.edu/assemblies/ontologies/woolly/2004/06/functions.owl#Process` entity will appear first in the results, with a little ‘heart’ next to it, indicating a hight trust value for the description of this particular entity (Figure 5.3). Other entities are also ranked at the top of the results, with a little ‘heart’, because of other statements reused by the user, or because of other evaluations provided by other users. More importantly, related entities like `http://edge.cs.drexel.edu/assemblies/ontologies/woolly/2004/06/functions.owl#Sense` are also assigned a higher trust value, because of the propagation the trust engine has operated from the positive evaluations assigned the `http://edge.cs.drexel.edu/assemblies/ontologies/woolly/2004/06/functions.owl#Process` (Figure 5.4).

5.4 Future Work

The work presented above is still in a preliminary stage. In addition to the preliminary tests we realized, further evaluations are required to check the usability of the system and the added value of community based trust evaluations for ranking in the Watson Plugin. Moreover, further developments and studies are required. In particular, the Watson Plugin currently only supports positive evaluations, while the Trust Engine can support negative trust evaluation which could be integrated in a feature to “report incorrect statements”.

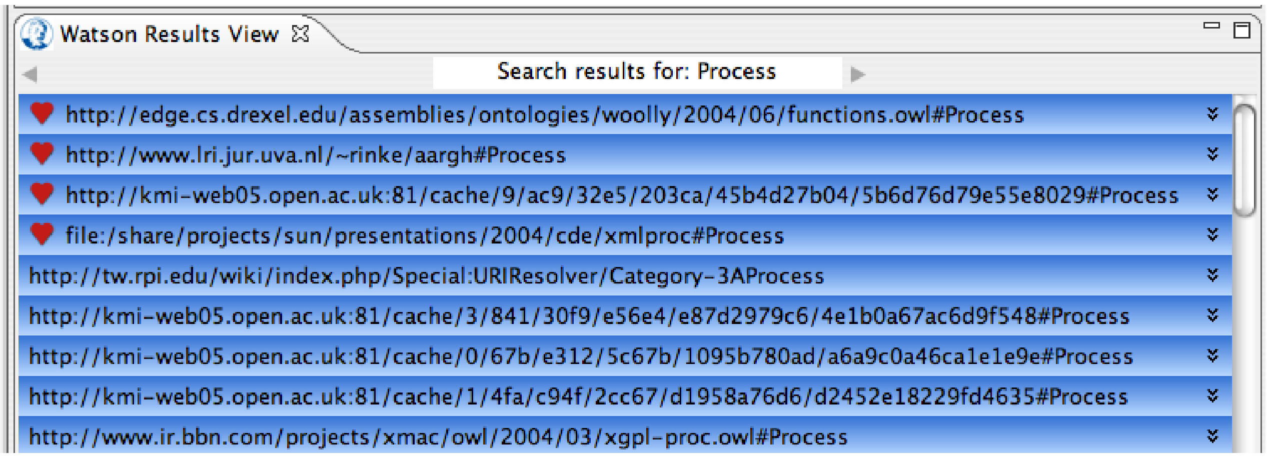
At a more general level, a common criticism of the sort of user evaluation provided here is that it ignores the possible influence of the context in which the user is using the statement in her evaluation. Indeed, the same user might find different statements correct or incorrect depending, for example, on whether she is building an ontology in the domain of medicine or one in the domain of theology. Integrating information about the


http://www.geneontology.org/GO-ont#Process
file:///share/projects/sun/presentations/2004/cde/xmlproc#Process
http://edge.cs.drexel.edu/assemblies/ontologies/woolly/2003/10/functions.owl#Process
http://edge.cs.drexel.edu/assemblies/ontologies/woolly/2004/06/functions.owl#Process

disjointWith	Indicate	Add relation from Process
disjointWith	Sense	Add relation from Process
subClassOf	Signal	Add relation from Process
Indicate	**disjointWith**	Add relation to Process
Sense	**disjointWith**	Add relation to Process

http://harth.org/2002/course#Process
http://kmi-web05.open.ac.uk:81/cache/e/558/85c8/95d05/bbaae1297b/83bcd8458f4c155f9#Process
http://www.ifomis.org/bfo/1.0/span#Process

Figure 5.2: A set of entities are retrieved for the class $Process$. The user integrate the statements number 2, 3 and 5 from the entity `http://edge.cs.drexel.edu/assemblies/ontologies/woolly/2004/06/functions.owl#Process`.



http://edge.cs.drexel.edu/assemblies/ontologies/woolly/2004/06/functions.owl#Process
http://www.lri.jur.uva.nl/~rinke/aargh#Process
http://kmi-web05.open.ac.uk:81/cache/9/ac9/32e5/203ca/45b4d27b04/5b6d76d79e55e8029#Process
file:/share/projects/sun/presentations/2004/cde/xmlproc#Process
http://tw.rpi.edu/wiki/index.php/Special:URIResolver/Category-3AProcess
http://kmi-web05.open.ac.uk:81/cache/3/841/30f9/e56e4/e87d2979c6/4e1b0a67ac6d9f548#Process
http://kmi-web05.open.ac.uk:81/cache/0/67b/e312/5c67b/1095b780ad/a6a9c0a46ca1e1e9e#Process
http://kmi-web05.open.ac.uk:81/cache/1/4fa/c94f/2cc67/d1958a76d6/d2452e18229fd4635#Process
http://www.ir.bbn.com/projects/xmac/owl/2004/03/xgpl-proc.owl#Process

Figure 5.3: The entity `http://edge.cs.drexel.edu/assemblies/ontologies/woolly/2004/06/functions.owl#Process` now appears first in the list of retrieved entities, with a little 'heart' indicating its high trust value.

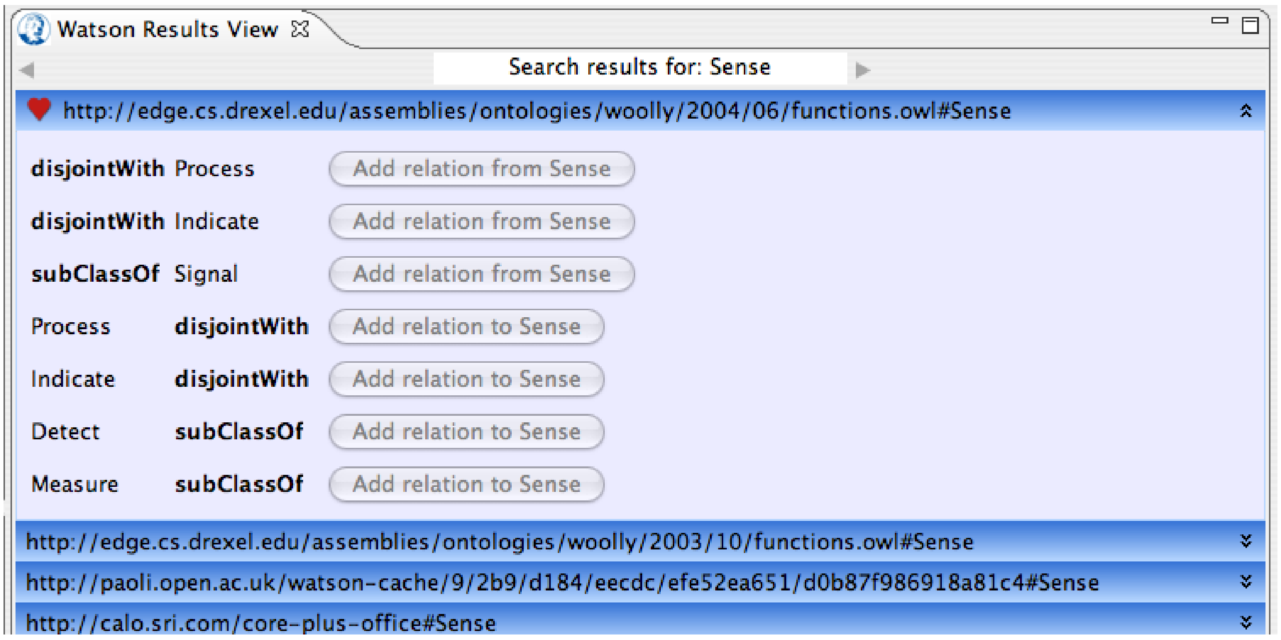

Watson Results View
Search results for: Sense
http://edge.cs.drexel.edu/assemblies/ontologies/woolly/2004/06/functions.owl#Sense
disjointWith Process
Add relation from Sense
disjointWith Indicate
Add relation from Sense
subClassOf Signal
Add relation from Sense
Process disjointWith
Add relation to Sense
Indicate disjointWith
Add relation to Sense
Detect subClassOf
Add relation to Sense
Measure subClassOf
Add relation to Sense
http://edge.cs.drexel.edu/assemblies/ontologies/woolly/2003/10/functions.owl#Sense
http://paoli.open.ac.uk/watson-cache/9/2b9/d184/eecdc/efe52ea651/d0b87f986918a81c4#Sense
http://calo.sri.com/core-plus-office#Sense

Figure 5.4: Trust values for other, related entities are also affected, like in the case of `http://edge.cs.drexel.edu/assemblies/ontologies/woolly/2004/06/functions.owl#Sense`.

context where an evaluated statement is to be used would then allow us to make the ranking even more "personalized" by also considering how similar the current context is with the contexts in which statements have been evaluated.

Finally, the current version of our Trust Engine only considers propagation of trust values within an ontology. However, as explained in Section 8.3, many different ontologies may cover the same domain and contain statements that either agree or disagree with each other. Considering such relations between ontologies could be a way to propagate trust values from users across ontologies, thus making a more efficient use of the collected quality information. Indeed, whenever a user finds a statement correct, she implicitly also find any statements agreeing with the evaluated one correct. Inversely, it is likely that when she would find incorrect statements that contradict the ones she evaluated positively. Using such principles could make possible to provide the user with quality information for particular ontologies based on the trust information she has expressed in other ontologies. Other mechanisms could also be used to take into account the 'network of users' within the trust engine, putting for example more weight on the evaluations of users who expressed similar opinions.

Part III

Methods for Evaluating Ontology Statements

Chapter 6

Introductory Notes

Ontology statements are the atomic components of all Semantic Web structures[1]. One of the distinguishing characteristics of the NeOn Toolkit and its associated tools is the reuse of ontology statements to support a variety of ontology tasks such as editing, matching, evolution or enrichment. For all these tools, it is essential to be able to predict the quality of the reused statements. In this chapter we give examples of tools that motivate research on evaluating ontology statements. We also present the research methodology that we have followed to investigate methods for statement evaluation.

6.1 Motivation

In this section we describe some of the tools that require statement evaluation methods.

The Watson NeOn Toolkit plugin[2] supports the ontology editing process by allowing the editor to reuse a set of relevant ontology statements drawn from online ontologies. Concretely, for a given concept selected by the user, the plugin retrieves all the statements in online ontologies that contain this concept (i.e., concepts having the same label). The statements are presented in an arbitrary order. Because of the typically large number of retrieved sentences it would be desirable to rank them according to their correctness. To date, however, no such methods exist causing the users some problems in finding the best statements.

Another core technology affected by this issue is Scarlet[3], an algorithm for deducing semantic relations from online ontologies. This technology provides a useful functionality that has been used to support a variety of ontology based tasks such as ontology matching [SdM08a, EdSZ07], folksonomy enrichment within the FLOR tool [Vil09] and ontology evolution in the Evolva NeOn Plugin [ZSdM08] . Unlike in the case of the Watson plugin, most of these tools automatically reuse the statements provided by Scarlet. Therefore, methods for filtering out the correct statements are even more important then in the case of the Watson plugin where a user is involved in the process.

Methods for evaluating ontology statements also have an applicability beyond the NeOn toolkit. For example, the OAEI series of evaluating ontology matching tools has traditionally focused on evaluating alignments containing equivalence mappings between the elements of two ontologies. However, as more and more matchers are capable of identifying other mappings than equivalence, the current gold-standard based evaluations need to be revised as it is impossible to manually predict all types of relations that would hold between the elements of two ontologies [SG08]. Our intention is to use the work presented in this deliverable as a potential solution to automatically evaluating complex alignments.

[1]We refer here to a statement in the sense of a knowledge triple rather than an RDF triple.
[2]`http://watson.kmi.open.ac.uk/editor_plugins.html`
[3]http://scarlet.open.ac.uk/

6.2 Methodology

The methodology followed to perform this line of work consisted of the following steps:

Understanding the problem. We started our work by investigating scenarios where suboptimal quality of statements caused a problem. We identified types of problematic statements from these cases. We will report on two such investigations in this section.

Acquiring test data sets. We acquired a total of six datasets containing statements used in two main tasks (ontology matching and evolution) and from two different domains (agriculture and scholarly publications). Each dataset contains a set of relation as well as their evaluations by multiple experts. The datasets are described in Section 6.3.

Identifying methods. We have taken an exploratory and experimental approach to identify methods for predicting statement correctness. We have identified a set of methods, some inspired from existing work in the field of NLP, and applied them to our data sets. We then fine-tuned them based on the obtained results.

Implementing methods in concrete tools. Finally, those methods that provided the best results were integrated in concrete NeOn tools.

Statement Type	Examples
Incorrect sense when the senses of the concept in the base and reused ontologies differ	$Event$ has a different meaning in an ontology about Sports and in one about Research $Article$ as a publication type anchored to $Article$ as a grammatical element
Containing irrelevant concepts i.e., concepts that do not make sense outside the context of their ontology	$Book \subseteq \_anon699$ $Pan \subseteq T$ $Pan \subseteq T$
Containing abstract concepts E.g., Thing, Object, Agent, Root	$Publication \subseteq Root$ $Publication \subseteq Object$ $Publication \subseteq Thing$
Obvious relations that do not bring additions to the base ontology	$X = X$ $X \subseteq X$
Modeling errors	$Chapter \subseteq Book$ $Article \subseteq Journal$

Table 6.1: Types of incorrect statements that should not be presented by the Watson plugin.

One of the investigations we performed to better understand the kind of incorrect statements that should be filtered was an experiment involving the Watson plugin. As detailed in deliverable D5.4.1., this experiment involved three experts that used the plugin to perform two task: ontology building from scratch and ontology refinement. Table 6.1 contains a summary of the type of problematic statements that were identified.

Another set of investigations were performed in the context of the Scarlet-based matcher. A manual evaluation of more than 200 incorrect statements (in the form of mappings between two ontologies), led to the identification of several categories of errors as those shown in Table 6.2 and explained briefly here. For each error type, we show the number of such errors in the evaluated set and the percentage it represents. We also provide examples for each error type, containing the URI and label of the source and target concepts, the established relation, as well as the path that lead to deriving this relation. The full details of the experiment can be found in [SdM08a].

Anchoring errors are a side-effect of the basic, string matching based anchoring and appear when a concept is related to an incorrect sense in online ontologies. For example, in the first entry in Table 6.2, concept *c_3179* describing "Game" in the sense of a hunted animal is incorrectly anchored to the *Game* concept in SUMO which represents a physical activity. In the second example, both concepts are anchored incorrectly. First, *c_6443* labeled with "Rams" and referring to an *"uncastrated adult male sheep"*[4] is put in correspondence with a similarly labeled concept ("ram"), but which refers to *Random Access Memory* in the context of the online ontology. In the same way, the *memory* concept of NALT refers to the term used as in psychology and thus has been incorrectly anchored to the identically labeled concept which refers to computer memory.

Subsumption used to model generic relations. One of the most common errors in online ontologies was the use of subsumption as a way to model the fact that there exists some type of relation between two concepts, e.g., $Survey \sqsubseteq Marketing$, $Irrigation \sqsubseteq Agriculture$, $Biographies \sqsubseteq People$.

Subsumption used to model part-whole relations. Subsumption is used in several ontologies to model part-whole relations. This resulted in incorrect mappings such as $Branch \sqsubseteq Tree$, $Leaf \sqsubseteq Plant$.

Subsumption used to model roles. Roles are often modeled as subclass relations, for example, that $Aubergine, Leek \sqsubseteq Ingredient$ ($Leek$ is a $Vegetable$ but in some contexts it plays the role of an ingredient).

Inaccurate labeling. We also found cases of correct subclass relations which introduced errors due to the inaccurate labeling of their concepts. For example, O_1[5] states that $coal \sqsubseteq industry$, where $coal$ refers to *coal industry* rather than the concept of $Coal$ itself. Similarly, for $Database \sqsupseteq Enzime$ in O_1[6], $Enzyme$ refers to an *enzyme database* rather than describing the class of all enzymes. Note that this type of errors could be avoided by a semantic, context aware anchoring mechanism.

Different Views. Finally, some of the explored ontologies adopted views that were not in concordance with the context of the mapping and/or the perspective of the evaluators. For example, TAP considers *lobsters* kinds of *Fishes*, a perspective with which none of the evaluators agreed.

6.3 Experimental Data Sets

Table 6.3 shows a summary of the main datasets used in our study.

6.3.1 AGROVOC/NALT Alignment

This data set has been obtained by performing an alignment between FAO's AGROVOC ontology and NALT. The relations are established between the concepts of the two ontologies and they are of three types: $\subseteq$, $\supseteq$ and $\perp$. Each relation has been evaluated by two experts. The details of this matching experiment are described in [SdM08a].

6.3.2 Extensions to AKT ontology

This dataset has been obtained as a side effect of testing a core component of the Evolva ontology evolution system. Evolva uses various background knowledge sources to evolve a given base-ontology. In a concrete experiment, described in [ZSdM08], we have evolved the AKT portal ontology with concepts that were

[4]Definition from WordNet2.1.
[5]`http://www.aifb.uni-karlsruhe.de/WBS/meh/mapping/data/russia1a.rdf`
[6]`http://mensa.sl.iupui.edu/ontology/Database.owl`

Error Type	Nr./ %	Examples				
		AGROVOC Concept	**Labels**	**Rel.**	**NALT Concept**	**Labels**
Anchor	114, 53%	*c_3179*	Game, Hunted Animals	$\xrightarrow{\sqsupseteq}$	*sports*	sports, ball games, athletics
		O_1:Game $\sqsupseteq$ O_1:Sport				
		O_1 = `http://lists.w3.org/Archives/Public/www-rdf-logic/2003Apr/att-0009/SUMO.daml`				
		c_6443	Rams, Tups	$\xrightarrow{\sqsubseteq}$	*memory*	memory
		O_1:ram $\sqsubseteq$ O_1:memory				
		O_1 = `http://www.arches.uga.edu/~gonen/qos_bilal.owl`				
Subsumption as generic relation	40, 18%	*c_3954*	Irrigation	$\xrightarrow{\sqsubseteq}$	*agriculture*	agriculture, agriculture (general)
		O_1:Irrigation $\sqsubseteq$ O_1:SoilCultivation $\sqsubseteq$ O_1:Agriculture				
		O_1 = `http://sweet.jpl.nasa.gov/ontology/human_activities.owl`				
Subsumption as part-whole	16, 7%	*c_23995*	Branches	$\xrightarrow{\sqsubseteq}$	*trees*	trees
		O_1:Branch $\sqsubseteq$ O_1:Tree				
		O_1 = `http://www.site.uottawa.ca/~mkhedr/NewFuzzy.owl`				
Subsumption as role	11, 5%	*c_6211*	Products, Produce	$\xrightarrow{\sqsupseteq}$	*wool*	wool
		O_2:Product $\sqsupseteq$ O_1:ManufacturedProduct $\sqsupseteq$ O_1:TextileProduct $\sqsupseteq$ O_2:Fabric $\sqsupseteq$ O_3:Wool				
		O_1 = `http://reliant.teknowledge.com/DAML/Economy.owl` O_2 = `http://reliant.teknowledge.com/DAML/SUMO.owl` O_3 = `http://reliant.teknowledge.com/DAML/Mid-level-ontology.owl`				
Inaccurate labeling	12, 5%	*c_1693*	Coal	$\xrightarrow{\sqsubseteq}$	*industry*	industry
		O_1:coal $\sqsubseteq$ O_1:industry				
		O_1 = `http://www.aifb.uni-karlsruhe.de/WBS/meh/mapping/data/russia1a.rdf`				
		c_24833	Databases, Data bases, Databanks	$\xrightarrow{\sqsupseteq}$	*enzymes*	enzymes
		O_1:Database $\sqsupseteq$ O_1:Enzyme				
		O_1 = `http://mensa.sl.iupui.edu/ontology/Database.owl`				
Different View	12, 5%	*c_2943*	Fishes	$\xrightarrow{\sqsupseteq}$	*lobsters*	lobsters
		O_1:Fish $\sqsupseteq$ O_1:MarineInvertebrate $\sqsupseteq$ O_1:Crustacean $\sqsupseteq$ O_1:Lobster				
		O_1 = `http://139.91.183.30:9090/RDF/VRP/Examples/tap.rdf`				

Table 6.2: Examples of several types of false mappings.

Data Set	Nr. of Relations	Type of Relations	Nr. or Evaluators
AGROVOC/NALT	737	$\subseteq$, $\supseteq$, $\perp$	2
AKT	80	$\subseteq$, $\supseteq$, $\perp$, named relations	3
OAEI'08 301	112	$\subseteq$, $\supseteq$, $\perp$, named relations	1
OAEI'08 302	116	$\subseteq$, $\supseteq$, $\perp$, named relations	1
OAEI'08 303	458	$\subseteq$, $\supseteq$, $\perp$, named relations	1
OAEI'08 304	386	$\subseteq$, $\supseteq$, $\perp$, named relations	1

Table 6.3: Overview of data sets.

automatically extracted from a corpus of relevant texts. The relations between the concepts of the base-ontology and the extracted concepts were identified by Scarlet. Besides the $\subseteq$, $\supseteq$ and $\perp$ relation types, this data set also contains named relations (e.g., $inJournal(Article, Journal)$). Each of the 80 relations in this data set has an assigned correctness value upon which 3 experts have agreed.

6.3.3 OAEI'08 3** alignments

The OAEI'08 dataset represents the alignments obtained by the Spider system on the 3** benchmark datasets and their evaluation [SG08]. This dataset contains four distinct datasets representing the alignment between the benchmark ontology and the MIT (301), UMBC(302), KARLSRUHE(303) and INRIA(304) ontologies respectively.

Chapter 7

Natural Language-based Techniques

Our general aim is to determine by means of different Natural Language (NL)-based techniques the correctness of ontology statements, that is, to validate ontology statements. Such validation consists in determining in a (semi-)automatic way if the relation expressed in ontology statements (e.g., subclass-of, part-whole, disjoint classes, etc.) is (formally) correct or not.

In the present chapter we attempt to offer an overview of the different approaches we have followed to validate ontology statements, which can be summarized in two. In a first approach, we have combined information directly retrieved from the Web by means of different techniques (corpora, Google) with linguistic patterns. In a second approach, we have relied on the WordNet lexicon. The objective pursued in both approaches is the possibility of automatically asserting whether a certain ontology statement is correct, or not, and in the latter case, to propose a correct ontology statement.

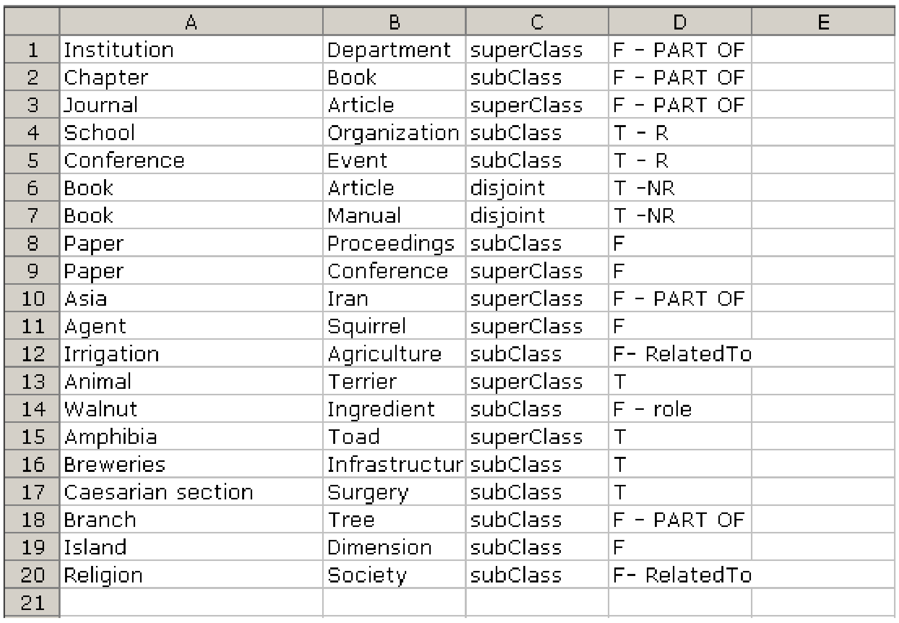
In the following sections we show the different phases in the *ontology statements validation* tests as they took place in time, and how the results from some tests motivated further tests, or brought us to discard some techniques and look for new ones. We have divided these validation tests in two main sections: Phase I, and Phase II.

7.1 Phase I

The idea of committing ourselves to the validation of ontology statements by means of NL-based techniques emerged as an intuition that some linguistic constructs can reliably convey the relations expressed in ontology statements. This thought has also been the basic motivation behind the research on Lexico-Syntactic patterns (LSPs), which is currently being conducted in the context of the NeOn project within T2.5 (see [PGD$^+$08], and [dCGPMPSF08]). The LSPs developed within NeOn can be defined as formalized representations of linguistic constructs or lexico-syntactic structures that correspond to a set of Ontology Design Patterns (ODPs), such as the ODP for the subclass-of relation, the ODP for the object property, the ODP for disjoint classes, or the ODP for the part-whole relation, among others. An initial repository of LSPs is available in [dCGPMPSF08]. This brought us to consider the possibility of taking advantage of the work done with LSPs for the validation of ontology statements. Therefore, the research we present in Phase I was carried out as a first informal trial to confirm our intuition or discard the option of validating ontology statements with the help of linguistic patterns.

7.1.1 Set of ontology statements used in Phase I

In this phase of the *validation tests* the starting point was an Excel file with a list of 20 ontology statements retrieved from the AGROVOC/NALT dataset. From the 20 ontology statements, 12 of them encoded erroneous information from a (formal) modelling perspective, i.e., the relation that was asserted to be holding between the ontology concepts was false (e.g., by the use of the subclass-of relation to model a part-whole relation). For the remaining 8 statements, the information was correct. This had been previously evaluated by



	A	B	C	D	E
1	Institution	Department	superClass	F - PART OF	
2	Chapter	Book	subClass	F - PART OF	
3	Journal	Article	superClass	F - PART OF	
4	School	Organization	subClass	T - R	
5	Conference	Event	subClass	T - R	
6	Book	Article	disjoint	T -NR	
7	Book	Manual	disjoint	T -NR	
8	Paper	Proceedings	subClass	F	
9	Paper	Conference	superClass	F	
10	Asia	Iran	superClass	F - PART OF	
11	Agent	Squirrel	superClass	F	
12	Irrigation	Agriculture	subClass	F- RelatedTo	
13	Animal	Terrier	superClass	T	
14	Walnut	Ingredient	subClass	F - role	
15	Amphibia	Toad	superClass	T	
16	Breweries	Infrastructur	subClass	T	
17	Caesarian section	Surgery	subClass	T	
18	Branch	Tree	subClass	F - PART OF	
19	Island	Dimension	subClass	F	
20	Religion	Society	subClass	F- RelatedTo	
21					

Figure 7.1: Excel file with the 20 ontology statements used in Phase I.

human evaluators, who, in the case of the erroneous statements, had additionally added information about the correct modelling solution.

The Excel file contained the following type of data, as can be seen in Figure 7.1:

Column A: the domain (concept) of the relation in the statement.

Column B: the range (concept) of the relation in the statement.

Column C: the relation automatically derived (that is, the relation included in the ontology statement).

Column D: a human evaluation of the statement (whether it is false (F) or true (T)). In case of a false statement, the correct relation is also included.

7.1.2 Description of the techniques applied in Phase I

In Phase I, our approach was centred in the use of corpora with the aim of retrieving sentences or parts of sentences (phrases) in which the terms corresponding to the ontology concepts in the ontology statements appeared related in some way. Then, the resulting corpora sentences or phrases were compared against LSPs. For this purpose, we selected those LSPs from the repository that involved in some way the relations (subclass, superClass, part-of, disjoint ...) expressed in the ontology statements in the Excel file Figure 7.1, namely:

- LSPs corresponding to subclass-of relation ODP
- LSPs corresponding to subclass-of relation and disjoint classes ODPs
- LSPs corresponding to subclass-of relation and simple part-whole relation ODPs
- LSPs corresponding to simple part-whole relation ODP
- LSPs corresponding to object property, datatype property, and simple part-whole relation ODPs

The template that describes the LSPs, the list of abbreviations and symbols used in the description of LSPs, and the set of LSPs that concern us in this case, have been included in Appendix A.

Once corpora sentences had been retrieved and compared to the sub-set of selected LSPs, two different situations appeared:

1. The relation holding between the ontology concepts in the corpora sentences had a match with one or several LSPs. Then, thanks to the correspondence between LSPs and Ontology Design Patterns, we could assert the type of relation existing between the concepts in corpora sentences (e.g., subclass-of, part-whole, disjoint classes, participation, etc.). The resulting relation was compared with the relation in the ontology statement, i.e., the relation in Column C of the Excel file (see Figure 7.1). The relation expressed by the LSP and the relation in the ontology statement could coincide or not. In case of coincidence, the ontology statement could be confirmed to be formally valid. In case of disagreement, the ontology statement could be said to be false, and a new relation could be proposed, namely, the ontology relation represented by the LSP.
2. The relation holding between the ontology concepts in the corpora sentences did not match any LSP. In this case, the validation test with LSPs had to be abandoned.

In order to better understand the whole process, consider ontology statement number 2 in the Excel file in Figure 7.1: *chapter_subClass_book*. Our knowledge of the world tells us that the relation holding between chapter and book is not a hyperonymy-hyponymy relation (subclass-of relation), but a relation between an object and its parts, i.e., a meronymy relation (part-of relation).

The first step consists of recovering sentences or phrases from corpora in which these two entities (*chapter* and *book*) appear together. The strategies employed for this step will be explained in more detail next. From the corpus search we could obtain phrases such as:

the most important chapter in this book
The first chapter of his book
this book includes a chapter on juggling
This book includes an interesting chapter on the history of

The second step consists in comparing the extracted sentences against the repository of LSPs. In fact, the last two sentences in our example would match the LSP corresponding to the subclass-of and part-whole relations (LSP-SC-PW-EN, see Table A.4 in Appendix A). This is an ambiguous pattern that requires further disambiguation. Additional techniques are being investigated in this sense, as for example, the use of WordNet. As a result of the disambiguation step, we could conclude that the sentences retrieved conveyed the meaning of part-whole relation.

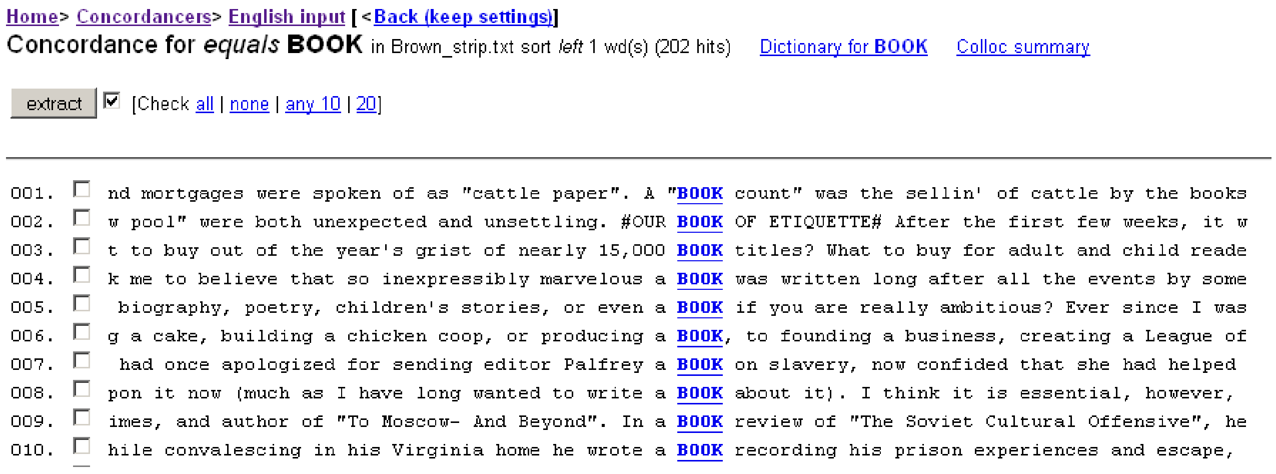
The third step would be to advert about the difference between the relation in the ontology statement (subclass-of), and the relation concluded from the comparison of the sentences against LSPs (part-whole). Finally, the idea would be to propose the relation in the LSP as the valid relation between the ontology concepts in the statement, i.e., the part-whole relation.

7.1.3 Set of corpora used in the validation tests

For conducting the tests based on corpora, we selected some of the most authoritative free online corpora that exist nowadays for the English language. A brief description of their main characteristics is included below:

British National Corpus - BNC [1]. This corpus contains a 100.000 million word collection of samples of

[1] http://www.natcorp.ox.ac.uk/


Home> Concordancers> English input [<Back (keep settings)]
Concordance for *equals* **BOOK** in Brown_strip.txt sort *left* 1 wd(s) (202 hits) Dictionary for **BOOK** Colloc summary

extract ☑ [Check all | none | any 10 | 20]

```
001. ☐ nd mortgages were spoken of as "cattle paper". A "BOOK count" was the sellin' of cattle by the books
002. ☐ w pool" were both unexpected and unsettling. #OUR BOOK OF ETIQUETTE# After the first few weeks, it w
003. ☐ t to buy out of the year's grist of nearly 15,000 BOOK titles? What to buy for adult and child reade
004. ☐ k me to believe that so inexpressibly marvelous a BOOK was written long after all the events by some
005. ☐  biography, poetry, children's stories, or even a BOOK if you are really ambitious? Ever since I was
006. ☐ g a cake, building a chicken coop, or producing a BOOK, to founding a business, creating a League of
007. ☐  had once apologized for sending editor Palfrey a BOOK on slavery, now confided that she had helped
008. ☐ pon it now (much as I have long wanted to write a BOOK about it). I think it is essential, however,
009. ☐ imes, and author of "To Moscow- And Beyond". In a BOOK review of "The Soviet Cultural Offensive", he
010. ☐ hile convalescing in his Virginia home he wrote a BOOK recording his prison experiences and escape,
```

Figure 7.2: Example of concordance lines from the Complete Lexical Tutor.

written and spoken English from a wide range of sources, and represents a wide cross-section of current British English, both spoken and written.

MICASE [2]. The MICASE corpus is a spoken language corpus of approximately 1.8 million words (200 hours) of contemporary university speech of the University of Michigan.

Complete Lexical Tutor [3]. This service gives access to different corpora, but one corpus has to be selected at a time. The BNC was also included in their repository of corpora, and since this had already been accessed, we selected the Brown corpus that contains 1.000.000 of words in English.

Cobuild Concordance and Collocation Sampler [4]. This Collins WordbanksOnline English corpus is composed of 56 million words of contemporary written and spoken text.

BWANANET [5]. BwanaNet is an interface developed at the IULA (Institut Interuniversitari de LinguŠstica Aplicada) that allows querying the Technical Corpus of the IULA Institute via Internet. This corpus contains documents in Spanish, Catalan and English on several subjects, such as medicine, computer science or law.

The way for searching corpora is by means of concordancers. A concordancer usually returns a display of 50 concordance lines where the word we are looking for appears accompanied by the words that precede and follow it. For an example of the first 10 concordance lines for the word book in the Complete Lexical Tutor see Figure 7.2. Some corpora allowed us to determine the distance between the target word and an additional word we wanted to precede or follow the target word. For example, with the Cobuild Concordance and Collocation Sampler, you could define a search like: *chapter+6book*. This would mean that the search engine would return only those concordance lines in which chapter appeared preceded or followed by the word book in a distance of up to 6 words.

7.1.4 Results of searching in corpora

We tried different complex search options with the British National Corpus (BNC), the Complete Lexical Tutor, and the Cobuild Concordance corpora. In the following sections we have included some examples of results from the BNC and the Cobuild Concordance corpora that are representative of type of results we obtained from corpora.

[2]http://quod.lib.umich.edu/m/micase/
[3]http://www.lextutor.ca/
[4]http://www.collins.co.uk/corpus/CorpusSearch.aspx
[5]http://bwananet.iula.upf.edu/indexes.htm

EX5 1219 Good reading notes taken by a good student will approximate to the structure of **the author’s chapter, book or article** –; they will include all key points but will omit the more detailed points

Figure 7.3: Concordace example for chapter_book from BNC.

GUR 1335 Conversations with two black pupils further revealed the dissatisfaction felt about the **school’s organization**

Figure 7.4: Concordace example for school_organization from BNC.

British National Corpus

The BNC permitted us to use the character “_" to match pairs of words that would appear in documents separated by one word. By using this search option, relevant results were only obtained for 6 out of the 20 ontology statements. As we can see from the examples, what we extracted were sentences in which the pair of concepts appeared related by means of certain conjunctions (Figure 7.3), the clitic s of the genitive case (Figure 7.4), prepositions (Figure 7.6) or separated by punctuations marks (Figure 7.5). This is not of much help if we consider that the LSPs to which we wished to compare the retrieved corpus sentences are verb centred.

Cobuild Concordance and Collocation Sampler

The Cobuild corpus allowed us to determine a greater distance between the word pairs we wanted to look up in the corpus up to 6 words. It was signalized in the following way: chapter+6book. In addition to this, with the aim of improving more results, we used a further strategy, which consisted in inverting the order of the word pairs that appeared in the Excel file (see Table 1). So, additionally to the chapter+6book pair, we introduced the book+6chapter pair. Results were then derived for 8 out of 20 ontology statements. Again, in most of the extracted sentences, the pair of concepts were related by means of prepositions (Figure 7.7), as nominal compounds (Figure 7.10), or they belonged to different sentences but were physically close to each other (last two sentences in Figure 7.8). On the other hand, we found some of these word pairs related by means of verbs, as in the three first examples of Figure 7.8, and four last examples of Figure 7.9. As previously mentioned, the word pairs related by means of verbs would be the only ones to be compared against LSPs in order to determine the type of relation holding between them. In “paper given to a recent conference" or “paper presented to a joint conference", examples included in Figure 7.9, verbs represent ad hoc relations that would not be recognized by LSPs.

7.1.5 Conclusions

From the results obtained from the corpora search, we can draw some interesting conclusions:

- In general, results obtained from corpora are quite scarce (we obtained results for less than 50% of all ontology statements). This makes it difficult to rely on this strategy for validating the correctness of

H8W 265 The Statement ‘;The Soviet Union refused to withdraw from Persia and put growing pressure on Turkey’;, describes how Stalin was desperate to secure the Soviet Union from attack from Europe and **Asia.Iran** (Persia) and Turkey were of great strategic importance to the Iron curtain countries because of their borders with Russia.

Figure 7.5: Concordace example for Asia_Iran from BNC.

HLA 2360 Later phases would concentrate on **irrigation for agriculture**.

Figure 7.6: Concordace example for irrigation_agriculture from BNC.

This is perhaps the most important **chapter** in this **book**, for without reading it you
question to which I shall return in the final **chapter** of this **book**, but his influence on a new
unique to social work. As Horne has argued in **Chapter** 5 in this **book**, whilst social work
by people of all social strata. The first **chapter** of his **book**, Woman of the Valley, describes

Figure 7.7: Concordance example for chapter+6book from Cobuild.

by Andrew Gelv, this **book** includes a **chapter** on juggling and unicycling. Most people
Francombe. This **book** includes an interesting **chapter** on the history of juggling and covers
the **book** is particularly strong. Abelson's **chapter** (with a third attack on BASIC) provides
than a graduation. The **book** finishes with a **chapter** on buying and chartering a boat. No
The Brothers Karamazov [f] Part II, **Book** V, **chapter** 4. [p] Eknath Easwaran, [f] The
40 years that one more **book** or even one more **chapter** could not add much that is new.
Flower [f] is a unique **book**. Beginning at **Chapter** XV (the Original Gate), it propounds
are the core concerns throughout this **book**. In **Chapter** 1 Nigel Parton focuses on the assu

Figure 7.8: Concordance example for chapter+6book from Cobuild.

a willingness to present a **paper** at the **conference** on our behalf but would need some help
section of the ballot **paper**. [p] [h] The **Conference** Arrangements Committee [/h] The
in the section of the ballot **paper** marked **Conference** Arrangements Committee. [p] Two
in the section of the ballot **paper** marked **Conference** Arrangements Committee - CLP
was presented in a **paper** given to a recent **conference** organised by the British Journal of
and Contiguity", **Paper** presented at a joint **conference** of the Association for the Study of
paper give to the AMED, CMED, Economist **Conference**, op. cit. Peters, T. J. and
in a background **paper** prepared for a Fabian **conference** at Oxford University earlier this

Figure 7.9: Concordance example for paper+6conference from Cobuild.

hardly believe that the title of a course or **conference paper** reveals everything about
was called [ZF1] the [ZF0] the s Commerce **Conference Paper**. And then occasionally for

Figure 7.10: Concordance example for conference+6paper from Cobuild.

ontology statements. In fact, some statements just show that the searched concepts appear physically close to each other, but there is no relation holding between them (see Figure 7.5).

- Some of the sentences retrieved show related concepts by means of verbs (Figure 7.8 and Figure 7.9), but these are the fewest. Therefore, not many sentences can be further employed for comparison against LSPs.
- Most of the examples show related concepts by means of prepositional modifications of nouns (Figure 7.6 and Figure 7.7). Some of them present a combination of concepts by means of nominal phrases with pre-modifying articles (Figure 7.10). We believe that these constructions are keener to appear in running text as such than the verbal constructs identified in LSPs. Therefore, we should consider using this kind of patterns for the purpose of exploiting corpora in addition to the LSPs.
- Some corpora returned no results at all. In the case of the MICASE corpus, we believe it is caused by the type of documents contained in the corpus, namely, transcriptions of conversations. This type of documents contains a great amount of interrupted or not completed sentences. Results were also very scarce in BWANANET. The reason for this may be the reduced number of documents contained in the corpus and its high level of speciality.

7.2 Phase II

Results from Phase I confirmed that our intuition about linguistic constructs reliably conveying the relations expressed in ontology statements was right. However, further strategies needed to be discovered and other techniques had to be used to improve results. In Phase II we decided to invert the steps followed in the previous strategy and make use of the Lexico-Syntactic patterns (LSPs) to obtain information from the Web, instead of extracting first the information from corpora and then comparing it to LSPs. Additionally, we made some preliminary tests with the well-known English lexicon WordNet. The whole process is detailed in the following sections.

The objective of this second phase of the validation tests was the same as in Phase I, namely, to find out if a certain ontology statement could be validated in an (semi)-automatic way relying on NL-based techniques.

In Phase II we decided to focus on a subset of all possible relationships between concepts. We selected the *subclass-of* and *part-whole* relations. The motivation behind this is that *subclass-of* and *part-whole* are ontology relations frequently mistaken by users when developing their ontologies (for a preliminary experiment on this see [dCGPMPSF08]).

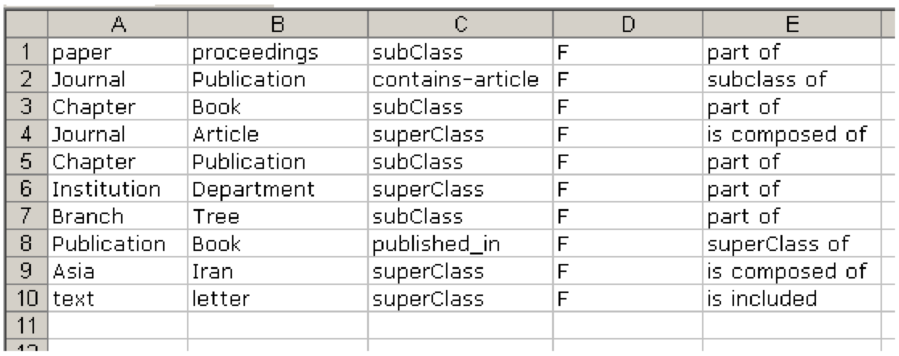
For this aim, we conducted two types of validation tests based on different NL-based techniques, as exposed in the following:

Technique 1. Linguistic patterns instantiation. This technique consist in:

1. Instantiating linguistic patterns for the subclass-of and part-whole relations with the ontology entities that appear in the statements of the file to be studied.
2. Manually introducing the resulting structures in Google.
3. Evaluating Google matches for the different linguistic patterns that correspond to the subclass-of or the part-whole relations, and establishing the validation of the ontology statements.

Technique 2. WordNet Comparison. The technique consists in deciding if the relation (subclass-of or part-whole) in the statement is correct or not based on the relation that exists between the two concepts involved in the ontology statement in the WordNet lexicon.

1. Automatically searching in the WordNet lexicon the pair of concepts appearing in the ontology statement. In this search, a possible relation existing between the two concepts is obtained.



	A	B	C	D	E
1	paper	proceedings	subClass	F	part of
2	Journal	Publication	contains-article	F	subclass of
3	Chapter	Book	subClass	F	part of
4	Journal	Article	superClass	F	is composed of
5	Chapter	Publication	subClass	F	part of
6	Institution	Department	superClass	F	part of
7	Branch	Tree	subClass	F	part of
8	Publication	Book	published_in	F	superClass of
9	Asia	Iran	superClass	F	is composed of
10	text	letter	superClass	F	is included
11					

Figure 7.11: Excel file with the 10 ontology statements used in Phase II.

2. Comparing the relation obtained in the search using WordNet with the relation in the ontology statement.
3. Establishing the validation of the ontology statement based on the previous comparison.

7.2.1 Set of ontology statements used in Phase II

In this second phase of the ontology statements validation tests, the input was an Excel file including a list of 202 ontology statements that had been retrieved from ontologies available in the Web. The list of ontology statements had been previously reviewed by ontology engineers, with the aim of checking the correctness of the statement from a (formal) modelling perspective. Ontology experts had to decide if the information encoded in the statement represented a reasonable modelling solution, or if it resulted in a modelling error (e.g., by the use of the subclass-of relation to model a part-whole relation).

We analyzed the list of the 202 statements and filtered them according to the following criteria:

- First, we discarded ontology statements that seemed having no sense in the reality or for our purposes (e.g. “book subclassof book", “informal superclassof misc", “organization subclassof organization", “collection superclassof organization", “collection superclassof person").
- Second, from those ontology statements that made sense, we chose those representing a false modelling solution and marked as F.
- Third, from those marked as false, we further selected those statements in which the encoded meaning was that of the subclass-of and part-whole relations.
- Forth, we analyzed the resulting statements and included the correct relation for each pair of concepts obtaining the Excel file shown in Figure 7.11 with the following information:

  Column A: the domain (concept) of the relation in the statement.

  Column B: the range (concept) of the relation in the statement.

  Column C: the relation automatically derived (that is, the relation included in the ontology statement).

  Column D: a human evaluation of the statement (whether it is false (F) or true (T)). In case of a false statement, the correct relation is also included.

7.2.2 Technique 1. Linguistic pattern instantiation

In general, this technique consists in instantiating different types of linguistic patterns with the terms representing the ontology concepts in ontology statements. The linguistic patterns we will take into account in this research can be generally defined as linguistic constructs in English that combine certain lexical elements in a specific syntactic order, and that permit to extract some information about the meaning they express (see also [MPdCGPSF08]). Specifically, in this Phase II of the validation tests we will use three different types of patterns: 1) the Lexico-Syntactic Patterns developed within NeOn in T2.5, 2) the so-called Hearst patterns, and 3) a set of patterns being used within the MUSING project for an automatic construction of ontologies from domain texts. A complete description of the different sets of patterns is included in more detail in the following, as well as the motivation behind its employment.

The first set of patterns consists of the Lexico-Syntactic Patterns (LSPs) being developed within NeOn. An initial excerpt of the repository of LSPs can be found in [dCGPMPSF08]. This repository consists of LSPs from regular NL expressions in English that have a correspondence with a sub-set of Ontology Design Patterns (ODPs). The objective of the LSPs research is to find the correspondences in NL to the relations expressed in ODPs in order to help untrained users in the reuse of ODPs. In LSPs, verbs are the ones that carry the semantic of the relation, as can be seen in the LSPs for the subclass-of and disjoint classes ODPs included in Table 7.1.

LSP Identifier	LSP-SC-Di-EN	
NeOn ODPs Identifier	LP-SC-01 + LP-Di-01	
Formalization	1	NP<superclass> be | CATV [either] NP<subclass> or NP<subclass>
	2	NP<superclass> be divide | split | separate | group in|into [either] [(NP<class >,)* and] NP<class>
Examples	1	*Animals are either vertebrates or invertebrates.*
	2	*Sensors are divided into two groups: contact and non-contact sensors.*

Table 7.1: LSP including *subclass-of* relation and *disjoint* classes ODPs.

Now, for Phase II we have selected those LSPs that have been identified for the **subclass-of** and/or the **part-whole** relations, since these are the relations expressed in the ontology statements we want to focus on. The current technique consist in manually instantiating those LSPs with the concepts in the ontology statements. It has to be noted that the correspondence between LSPs and ODPs is not always one-to-one, but one-to-two, or even one-to-many, as explained in more detail in [dCGPMPSF08]. This brought us to consider all LSPs that were in some way or the other related with the subclass-of and the part-whole relations, no matter if it was in a direct correspondence (1 LSP corresponds to 1 ODP) a one-to-many correspondence (1 LSP corresponds to N ODPs), or in a one-to-two correspondence (1LSP corresponds to the combination of N ODPs).

In Appendix A we have included the template that describes LSPs, as well as the 5 LSPs that come into consideration, which are the same as in Section 7.1.

Additionally to the verb-oriented LSPs we wanted to include other patterns that also take lexical and syntactic properties into account, but that normally stay under sentence level, i.e., they are constituted by nominal or prepositional constructions. The introduction of additional types of patterns in this phase of the validation tests has been motivated by the results obtained in Phase I (see Section 7.1), in which most of the results pointed to this kind of constructions. We have classified these patterns into two groups: the so called Hearst patterns, and the MUSING patterns. Within the first set of patterns, we include some of the well-known patterns identified in [Hea92] by Marti A. Hearst, who was the first author to apply lexico-syntactic patterns to the automatic extraction of related information from text. The second set of patterns has been extracted in the context of the European project MUSING[6] with the aim of semi-automatically deriving ontologies from domain texts [DV06, VD09]. The tables below summarize both sets of patterns:

Regarding Hearst patterns, it has to be noted that in order to extract the meaning we are looking for of subclass-of relation, the comma between the superclass and the preposition is crucial, and Google does not consider it when matching strings. This means that the matches for “publications including journals" and “publications, including journals" are both counted as the same, when the corresponding meanings are

[6] www.musing.eu

Hearst patterns	HYPERNYM-HYPONYM RELATION	
Formalization	1	NP_0 such as NP_1, NP_2 . . . . (and / or) NP_n
	2	NP, , including NP* or/and NP
	3	NP, , especially NP* or/and NP
	4	NP , NP* , and other NP
Examples	1	*The bow lute, such as the Bambara ndang...*
	2	*All common-law countries, including Canada and England...*
	3	*...most European countries, especially England, Spain, and France...*
	4	*...temples, treasuries, and other important civic buildings...*

Table 7.2: Hearst lexico-syntactic patterns for the hypernym-hyponym relation from [Hea92]

completely opposed. The instantiation without commas would have the meaning of a part-of relation, referring only to those publications that include journals as parts of them and not any other kind of publications. Whereas, when putting a comma between “publications" and “including", it would mean that “journals" are one of the many types of listed publications. This makes results not so reliable.

MUSING patterns	PART-OF RELATION	
Formalization	1	Noun<part> of | in Noun<whole>
	2	Noun<whole> + Noun<part>
Examples	1	*...chief of the corporation...*
	2	*...bank employees...*

Table 7.3: MUSING string-based patterns from [DV06, VD09]

Regarding MUSING patterns presented in Table 7.3, we have to remark that only when the patterns in 1 (*Noun<part> of Noun<whole>* and *Noun<part> in Noun<whole>*) that correspond to a post-nominal modification, produce relevant matches, results of the whole set will be taken into account, otherwise results will be discarded. The reason for that is that the second nominal compound pattern *Noun<whole> + Noun<part>*, is quite ambiguous and can correspond to other relations besides the **part-whole relation**. For example, “conference paper" would not indicate a part-whole relation holding between conference and paper, but it is a type of paper. Therefore, the patterns in 1 will be used as a kind of validation for pattern 2, as already proposed by the authors in [VD09]. Another point that has to be made here is that the first noun in the nominal compound pattern *Noun<whole> + Noun<part>* has to be a singular noun. This will be the only exception to the use of the plural form in generalizations, and therefore, when talking about classes. Additionally, by the application of pattern 2 we encounter the same problem as by Hearst patterns, namely, Google does not take into account if there is a comma, a full stop, a hyphen or any other punctuation mark between the two nouns, and counts all matches were the two nouns are physically next to each other.

Our intuition is that these two groups of patterns, Hearst and MUSING patterns, may appear more frequently in running text because of various reasons. In the first place, they have been directly extracted form available documents in the Web. In the second place, the constructions they contain are 1) restricted to specific phrases that do not normally admit modifications or the inclusion of additional elements, and 2) do not demand the existence of a full sentence construction. Therefore, it is more probable that they are word-for-word matched in texts than full sentences. For example, it is easier to match “bank employees" in Google than “banks have employees", which may appear with an infinite number of variants in the Web, such as “banks have faithful employees", “banks in England have many employees that (...)", “banks do not have rude employees", etc.

The main reason for including this sort of patterns in our tests is that they are more probable to appear in web documents than LSPs. This was also confirmed in Phase I of the validation tests (see Section 7.1.4). However, we also believe that the lack of textual context may impose some limitations on them and cause the identification of irrelevant or false relations that will be counted in the total amount of Google appearances. This will have to be considered when evaluating the results. On the other hand, we expect the identification of more reliable examples on the part of the LSPs, since they rely on whole sentential constructions and the possibilities of ambiguities are enormously reduced. Thus, our hypothesis can be summarized in these two ideas:

- Hearst and MUSING patterns return a higher number of matches in Google, but irrelevant or erroneous relations are likely to be counted.
- LSPs patterns return a lower number of matches in Google, but the matched relations are more likely to be correct or valid.

Patterns instantiation and application of the technique

In this section, our aim is to describe how we proceeded with the manual instantiation of LSPs and their subsequent search in Google. Let us take as example the ontological statement *paper_subclass_of_proceedings* considered FALSE or incorrect by human evaluators in the Excel file shown in Table 7.1.

The names of the concepts “paper" and “proceedings" will be now used to instantiate the different LSPs in which the **subclass-of relation** and the **part-whole relation** are involved. One LSP for the **subclass-of relation** corresponding to LSP-SC-EN (Table A.2) is:

NP<subclass> be [CN] NP<superclass>

The instantiation would result in a sentence in NL like this: *papers are proceedings* or *papers are types of proceedings*, among many other options. Note that we have changed paper for its plural form papers. This is needed because we are referring to classes, and the English language usually uses the plural form for substantives when generalizing and referring to classes that include individuals. Therefore, we will use the plural form of terms. We have also seen that some elements in the same patterns can be instantiated in multiple ways: *papers are proceedings, papers are types of proceedings, papers are subclasses of proceedings, papers are members of proceedings,* etc. In cases like this, we have decided to go for the basic option, leaving out the optional elements, in our example [CN], which stands for: *Class Name. Generic names for semantic roles usually accompanied by preposition, such as class, group, type, member, subclass, category, part, set, example etc* (as detailed in Table A.7 of Symbols and Abbreviations used in LSPs). The reason for using the pattern in its basic composition excluding optional elements is motivated by the effort that the manual instantiation of LSPs with the optional elements would require.

The semantic role in the relation has to be maintained as well, so that *papers* <subclass> is on the left-hand side of the verb, and *proceedings* <superclass> on the right-hand side. When we instantiate the **part-whole relation** LSPs with these concepts, the semantic role of <subclass> in the **subclass-of relation** will become the <part> in the **part-whole relation**, and the <superclass> will be the <whole>. Taking the previous example, papers will now be considered parts of proceedings, and will instantiate the corresponding LSPs for the part-whole relation.

NP<whole> include | consist of | have | contain [CN] [PARA] (NP<part>,) and NP<part>*

The instantiation will then look like: *proceedings include papers; proceedings consist of papers; proceedings have papers; proceedings contain papers*. Here again, we will leave the optional elements out, but will instantiate the pattern with all alternative verbs for expressing the relation, since it implies a higher probability of obtaining results.

Once we have instantiated all LSPs in which the **part-whole relation** and **subclass-of relation** are involved, we proceed to manually include the resulting NL sentences into the Google search engine. The Google configuration options used in this step were 3:

1. It should only find those pages that contained the exact wording or phrase
2. It should only look in pages in English

3. It could access files in any format

The first configuration option is the most important, because in this way Google only counts the number of appearances in which web documents exactly contain the resulting NL sentences.

Validation test and results

In the present section, our objective is to give an overview of the results obtained by means of instantiating the different types of patterns, and then searching in Google for the instantiations. We will start by describing the table designed for the patterns instantiation technique that is to be seen in Table 7.4.
The table is divided in three columns:

- The column on the left corresponds to the type of patterns, i.e., LSPs for the subclass-of relation, Hearst patterns, and so on.
- The column in the middle displays the instantiated patterns with the terms representing the ontology concepts from the corresponding ontology statement.
- The column on the right offers the results from the matches returned by Google.

Then, the rows have been grouped in 5 sets, each of them comprising the patterns of a certain type (LSPs or Hearst patterns, or MUSING patterns) and representing a certain relation (e.g. the 1st set of rows comprises the subclass-of and subclass-of plus disjoint classes relations). Each set of rows is followed by a partial count.

- The 1st set of rows contains those patterns for the subclass-of relation, or subclass-of plus disjoint classes relation. The 1st partial counts their results all together.
- The 2nd set of rows contains Hearst patterns. These patterns also express the subclass-of relation, but the results obtained from their instantiation are counted on their own.
- The 3rd set of rows contains ambiguous LSPs, i.e., those patterns that can express both the subclass-of and the part-whole relation depending on their context. These are counted in the 3rdpartial.
- The 4th set of rows contains the patterns expressing the part-whole relation, or the part-whole, object property and datatype property relations depending on modelling decisions. Results are counted on the 4th partial count.
- The 5th set of rows belongs to MUSING patterns. Results will be just counted in the 5th partial when the three of them obtain significant results, otherwise they will be discarded.

Once we have described the template table designed for presenting the results of this technique, we will now look into more detail in the concrete results. Let us consider ontology statement number 3 in our Excel table (see Figure 7.11), *chapter_subclass_Of_book.* Our knowledge of the world tells us that a chapter is not a type of book, but a subdivision or a part of a book. The question now is if this conclusion can be drawn in a (semi-)automatic way by relying on linguistic pattern instantiations and their search in Google. We will analyze the concrete results for each type of patterns.

1st set of rows: LSP-SC-EN and LSP-SC-Di-EN. According to Google results, 12 matches have been found for LSPs that correspond to the subclass-of relation (and subclass plus disjoint classes relation). Considering instantiation 1.1, “chapters are books", it is difficult to imagine a sentence of this kind making sense in any text. However, there are sentences that make sense among the results, as for example, “Some chapters are books unto themselves". Obviously, sentences as such do not report

Patterns	Instantiations with ontology statement entities		Google Matches
LSP-SC-EN	1.1	“chapters *are* books"	7
	1.2	“chapters *are grouped into* books"	4
	1.3	“chapters *fall into* books"	0
	1.4	“chapters *belong to* books"	1
	1.5	“*examples* of books are chapters"	0
LSP-SC-Di-EN	1.6	“books *are classified into* chapters"	0
1st PARTIAL			**12**
Hearst patterns	2.1	“books, *such as* chapters"	10
	2.2	“books, *including* chapters"	380
	2.3	“books, *especially* chapters"	37
	2.4	“chapters, *and other* books"	27
2nd PARTIAL			**454**
LSP-SC-PW-EN	3.1	“books *include* chapters"	280
	3.2	“books *consist of* chapters"	19
	3.3	“books *are divided into* chapters"	1890
3rd PARTIAL			**2189**
LSP-PW-EN	4.1	“chapters *are contained* in books"	0
	4.2	“chapters *compose* books"	0
	4.3	“chapters *make up* books"	1
	4.4	“chapters *are part of* books"	2
	4.5	“books *are made up* of chapters"	2
	4.6	“books *contain* chapters"	547
LSP-OD-DP-PW-EN	4.7	“books *have* chapters"	3690
4th PARTIAL			**4242**
MUSING patterns	5.1	“chapters *in* books"	487000
	5.2	“chapters *compose* books"	0
	5.3	“book chapters"	3.780.000
5th PARTIAL			**4343500**

Table 7.4: Example 3: Ontology Statement- chapter_subclass_Of_book

about the nature of books, but rather express a metaphorical sense. Although we have to be conscious of the existence of such sentences, 7 matches is still a very small quantity for the whole Web.

Instantiation example 1.2 “chapters are grouped into books" returned 4 matches. In this case we find sentences like: *The chapters are grouped into books, each named for one of the Anishinabe seasons*. Although “to group into" is included here with its meaning of “to classify" as in *Perfumes are grouped into fragrance families*, it can also mean “to put together in groups", as in *These questions are grouped into blocks of three or more*. This may bring us to conclude that “to group into" could be considered an ambiguous pattern to be included in the 3rd row: LSP-SC-PW-EN, although this still needs more evidence.

Finally, we want to comment on the last example found for instantiation 1.4 “chapters belong to books". In this case we find only one example: *These chapters belong to books in the Bible,(...)*. This is also a very specific case, since some works are divided into books like the Bible. But, again, we are confronted with a very specific example.

2nd set of rows: Hearst patterns. By instantiating Hearts patterns we observe that the number of matches found increases considerably in comparison with the first set of patterns. However, none of the instantiations coveys the meaning of a subclass-of relation, as expected by the employment of these patterns. Consider the following examples:

- (...) browse the OLL website selecting entire books or parts of *books (such as chapters* or sections) for inclusion on the reading list;
- The sub-panel anticipates the majority of work submitted will take the form of articles, *books (including chapters* in books), working papers and monographs ...
- *Books (especially chapters* and early readers)...
- These form a system of elaborate cross-referencing to other *chapters and other books* resulting in a crowded layout at times.

As we can see, Google returned matches without respecting the use of the comma, which implies that none of the matches expresses the wished meaning. In fact, the first two examples refer to chapters as parts of books, and not as types of books. Therefore, we should be cautious about the resulting 454 matches, the reliability of which is not guaranteed.

3rd set of rows: LSP-SC-PW-EN. By instantiating this ambiguous LSP that can express subclass-of or part-whole relations at the same time, Google returned more than 2000 matches. This makes sense since the sentences resulting from the instantiation are quite usual ways of conveying the meaning of objects and their parts. Take a look at the sentences below:

- At a more detailed level, both *books include chapters* on the biological background for gene expression analysis (...).
- (...) some *books consist of chapters* written by a number of contributors.
- Each volume is divided into a number of books and the *books are divided into chapters* and the chapters are then divided into Ahadith (traditions).

They all express the meaning of part-whole and make sense as phrases as such without additional context. However, these patterns require further disambiguation, since they could also convey the subclass-of relation.

4th set of rows: LSP-PW-EN and LSP- OD-DP-PW-EN. Patterns for the part-whole relation include 7 different types of instantiation possibilities. From all the instantiation types, two of them are the most productive ones, namely, “books contain chapters", and “books have chapters". The rest of them produce scarce results. “To have" is the most general verb and the hypernym of the rest of verbs, and this may be the reason for the high number of matches. “Contain", “compose", make up" and so on, are more specific, and their use will be conditioned by the context. For example, if we talk about recipients or containers that hold something within themselves, “contain" or “make up" will be probably preferred. In the same way, “compose" will be rather used to refer to objects and their components. Nevertheless, we find valid examples for nearly all instantiations, as can be observed in the following sentences:

- Songs make up albums as *chapters make up books*.
- (...) we find that *chapters are part of books*, paragraphs are part of chapters, exercises are part of chapters, and so on.
- Lots of academic *books are made up of chapters* by different people, with another person or other people as editor.
- Both *books contain chapters* on Communities of Practice in Universities and Schools as well as Communities of Practice in Adult and Continuing Education.
- Both of these *books have chapters* dedicated to specific languages.

The low figures for instantiations 4.1 to 4.5 can be justified by the specificity of the verbs, and by the restrictions imposed by the elements in sentences or phrases.

5th set of rows: MUSING patterns. MUSING patterns are highly productive, as already foreseen, because the structures they represent are very common in language (by linking the pair of concepts by means of prepositions, or as nominal compounds). However, it cannot be asserted that whenever two terms appear related by means of these prepositions, the relation holding between them is a part-whole relation. Among sentences like *A chronological listing of chapters in books (as opposed to entire volumes)* ... that makes sense linguistically speaking, we find other headlines like HUMA-BOOK-CHAPTERS-HKUST-INSTITUTIONAL-REPOSITORY, where it is more difficult to assert the validity of the construction. All in all, the number of matches surpasses the results obtained for the rest of patterns. In those cases where the part-whole relation is the one valid between the ontology concepts, as in the case of chapter and book, matches can easily achieve the hundreds of thousands or even millions. In cases where the valid relation between the pairs of terms is a subclass-of relation, as e.g. between journal and publication (see Table 7.5), the number of matches is still considerable, but never so high.

Briefly, we will now refer to the results obtained for the ontology statement number 2 *journal_contain-article_publication* in the Excel file (Table 7.1). Contrary to our previous example, *chapter_subclass_Of_book*, our knowledge of the world tells us that “journal" is not a part of a publication, but rather a type or a subclass of the class “publication". As in the case of *chapter_subclass_Of_book*, we have included the table with the instantiations and results from Google below, see Table 7.5. The relation holding between the terms “journal" and “publication" representing the ontology concepts in the ontology statement *journal_contain-article_publication* should be a subclass-of relation, as pointed out by the ontology expert, and not a part-whole relation, as returned by the statement. If we just compare the results from the LSPs for the subclass-of relation, (1st partial: 442 matches), against the results from the LSPs for the part-whole relation (3rd partial: 2 matches), we can immediately conclude that the relation holding between journal and publication should be subclass-of. This finds a parallelism in the results from the Hearst patterns for the subclass-of relation, against the ones from the MUSING patterns for the part-whole relation. In the case of *journal_contain-article_publication*, the matches for Hearst patterns are 34611, against 5570 matches from the MUSING patterns. In view of the results, we could conclude that the relation holding between “journal" and “publication" is in fact a subclass-of relation.

Results were so clear cut for 8 out of the 10 ontology statements in the Excel file (Table 7.1): *paper_part-of_proceedings, journal_contains-article_publication, chapter_subClass_book, institution_superClass_department, branch_subClass_tree, publication_published-in_book, Asia_superClass_Iran*. However, for the remaining 2 ontology statements (*chapter_subClass_publication*

Patterns	Instantiations with ontology statement entities		Google Matches
LSP-SC-EN	1.1	“journals are publications"	442
	1.2	“journals *are grouped into* publications"	0
	1.3	“journals *fall into* publications"	0
	1.4	“journals *belong to* publications"	0
	1.5	“*examples* of publications are journals"	0
LSP-SC-Di-EN	1.6	“publications *are classified into* journals"	0
1st PARTIAL			**442**
Hearst patterns	2.1	“publications, *such as* journals"	1060
	2.2	“publications, *including* journals"	2870
	2.3	“publications, *especially* journals"	81
	2.4	“journals, *and other* publications"	30600
2nd PARTIAL			**34611**
LSP-SC-PW-EN	3.1	“publications *include* journals"	288
	3.2	“publications *consist of* journals"	2
	3.3	“publications *are divided into* journals"	0
3rd PARTIAL			**290**
LSP-PW-EN	4.1	“journals *are contained* in publications"	0
	4.2	“journals *compose* publications"	0
	4.3	“journals *make up* publications"	1
	4.4	“journals *are part of* publications"	0
	4.5	“publications *are made up* of journals"	1
	4.6	“publications *contain* journals"	0
LSP-OD-DP-PW-EN	4.7	“publications *have* journals"	1
4th PARTIAL			**2**
MUSING patterns	5.1	“journals *in* publications"	20
	5.2	“journals *compose* publications"	1010
	5.3	“journal publications"	4540
5th PARTIAL			**5570**

Table 7.5: Example 2: Ontology Statement- journal_contain-article_publication

and *text_superclass_letter*) the results obtained from the linguistic patterns instantiation did not coincide with the relation stated by the expert. In both cases, it was expected that Google results would be in favour of a part-whole relation, but in both, results supported the choice of subclass-of relation.

All in all, we can conclude that this technique got the right ontology relation in 80% of the cases. Nevertheless, it seems risky to us to make this affirmation without conducting these tests with a higher number of examples.

Conclusions

From the results of the linguistic patterns instantiation we can draw some interesting conclusions that can be summarized in 2 points:

- Our first hypothesis that Hearst and MUSING patterns return a higher number of Google matches than the LSPs has been confirmed by the results from the instantiations with the 10 ontology statements in the Excel file (Table 7.1). The main reason for this is that Hearst and MUSING patterns stay under sentence level -they are nominal or prepositional constructions- and are more probable to appear as such in documents. LSPs correspond to whole sentences and it is more difficult to find an exact correspondence in Web documents.

- Our second hypothesis regarding LSPs matches being more likely to correctly convey the expected relation than Hearst and MUSING patterns has also been confirmed on the light of the detailed analysis of the retrieved sentences for the statements *chapter_subclass_Of_book* and *journal_contain-article_publication*. We have also observed that some of the LSPs produce significant results, whereas others produce very scarce results (E.g., LSP 1.5, “examples of books are chapters" and 1.6, “books are classified into chapters" produced not results at all for the whole set of ontology statements).

7.2.3 Technique 2. WordNet Comparison

As already mentioned, this technique consists in deciding if the relation in an ontology statement -in our case we focus on subclass-of and part-whole relations- is correct or not by comparing it with the relation that exists in the WordNet lexicon between the two concepts involved in the ontology statement.

In applying this technique, the first step consists of automatically searching in WordNet the pair of concepts appearing in the ontology statement, thus obtaining as a result the relation existing in WordNet between the two concepts.

To carry out this search we used the Semantic Mapper algorithm [DdB$^+$08] that is capable of discovering and generating automatically concept specifications, attributes and relations of a conceptual model, an ontology, according to the elements of a relational model. Although this process is automatic, the results will not be completed or directly exploitable; besides, because of its heuristic nature, it should be supervised and validated by any user with a deep knowledge of the semantics underlying the models. The Semantic Mapper has two main components: a linguistic component and a structural one. In this case we are interested in the first one, the linguistic component, which detects lexical relations between the terms to infer semantic relations from them. To detect this type of relations the Semantic Mapper algorithm uses WordNet.

The main idea behind the Semantic Mapper algorithm is that it is possible to forecast the semantic relation between elements pertaining to two models from the lexical relation existing between the terms that represent them. It is quite sensible to think that when designing a model, the elements that compose such a model should have comprehensible and meaningful names instead of random names or codes since the latter complicate the management and use of the model. In other words, the names of the elements of a model (relations, attributes, concepts, etc.,) have detectable semantic "clues" that permit inferring the semantic relations existing between them. For example, a synonymy relation between two terms would probably hide a conceptual relation of equivalence.

The identification of the general semantic relations between each pair elements (concepts) involves applying a set of rules for implementing the notion of the lexical semantic correlation. For this purpose, the proposal described in [BSZ03] can be used, though any other proposal could be equally valid[7].

- **Equivalence ($=$):** Two “concepts" [A] and [B] are equivalent if at least one synset is common to [A] and [B]. In other words, it is very likely that two elements are equivalent if their names are synonyms.
- **Less general ($\subseteq$):** a “concept" [A] is less general than other concept [B] if there is a relation of hyponymy (hypo) between any synset of [A] and [B]. In other words, it is very likely that element [A] is less general than element [B] (there is an implication between A and B) if the term used to name concept [A] is a hyponym of the term used to name concept [B] (and the same occurs for the opposite case).
- **More general ($\supseteq$):** A “concept" [A] is more general than a concept [B] if there exists a relation of hyperonymy between any synset of [A] and any of [B].
- **Disjuncts ($\perp$):** Two concepts [A] and [B] are disjuncts if there is a relation of antonymy (ant) between any synset of [A] and any of [B]. In other words, it is very likely that two elements whose names are antonyms be disjuncts.
- **Overlapping ($\bigcap$):** This is the more general case, in which none of the previous relations can be confirmed because WordNet does not provide any relation between two “concepts". Two elements, whose names do not have any of the above relations, will usually have some degree of undetermined intersection.

We automatically applied the Semantic Mapper algorithm to the set of ontology statements presented in Section 7.2.1. As a result we obtained that there was no relation (overlapping) between the different pairs of concepts in the set of ontology statements.

The results obtained were surprising because when manually accessing WordNet online, some pairs of terms representing the concepts in the ontology statements were in fact holding a relation. In Figure 7.12 we can see that the terms representing the concepts journal and publications from the Excel file in Table 7.1 are in a hyponym- hypernym relation, or what is the same, journal is subclass of publication.

In view of the unproductive results, we could not perform the second and third steps, which consist respectively on 1) comparing the relation obtained in the search using the Semantic Mapper algorithm with the existing relation in the ontology statement and 2) on establishing the validation of the ontology statement based on the previous comparison.

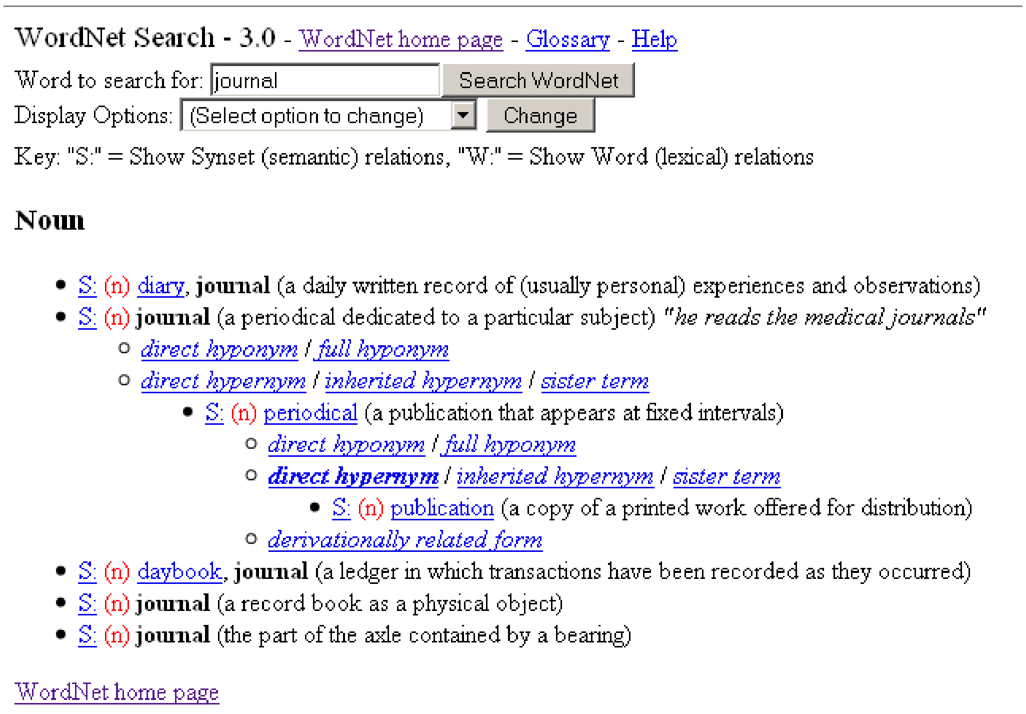
According to the manual confirmation that WordNet reliably relates terms by means of different types of relations, specially the ones we are interested in of subclass-of (hyperonymy-hyponymy) and part-whole (meronymy), we plan to analyze the Semantic Mapper algorithm in order to improve it and conduct some more validation tests in the future.

7.3 Conclusions and future lines of work

After the preliminary ontology statements validation tests presented in this chapter, we can extract some general conclusions that can guide us in further validation tests in the future. In general, it can be stated that the use of linguistic patterns for the validation of ontology statements has proven satisfactory on the light of the initial tests. Regarding the use of the Semantic Mapper algorithm to automatically validate relations in ontology statements by accessing the WordNet lexicon, results have not been the expected ones. For this reason, we plan to analyze the Semantic Mapper in more detail and provide some improvements that allow us a successful exploitation of the potential possibilities we believe WordNet can still offer.

Specifically, we already foresee some new approaches or the improvement of the current ones to tackle the problem of the ontology statements validation. We have tried to summarize them in the following:

[7]The use of WordNet provides satisfying results when creating semantic relations between simple concepts; however, these relations can be equally implemented if they are based on a set of domain ontologies or on any other resources that permit generating such relations through the application of any type of heuristic.


WordNet Search - 3.0 - WordNet home page - Glossary - Help

Word to search for: journal Search WordNet

Display Options: (Select option to change) Change

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Noun

- S: (n) diary, **journal** (a daily written record of (usually personal) experiences and observations)
- S: (n) **journal** (a periodical dedicated to a particular subject) *"he reads the medical journals"*
  - *direct hyponym* / *full hyponym*
  - *direct hypernym* / *inherited hypernym* / *sister term*
    - S: (n) periodical (a publication that appears at fixed intervals)
      - *direct hyponym* / *full hyponym*
      - ***direct hypernym*** / *inherited hypernym* / *sister term*
        - S: (n) publication (a copy of a printed work offered for distribution)
      - *derivationally related form*
- S: (n) daybook, **journal** (a ledger in which transactions have been recorded as they occurred)
- S: (n) **journal** (a record book as a physical object)
- S: (n) **journal** (the part of the axle contained by a bearing)

WordNet home page

Figure 7.12: WordNet capture of the relation between the terms "journal" and "publication".

1. Despite of the results from corpora not being as good as expected in the first trial, we would still like to look for more strategies to exploit corpora possibilities. In this sense, we plan to conduct some trials with the **Web Corp**[8] search engine, a tool designed to search in the whole web for concordances in real time making use of Google, Altavista, etc. This search engine takes advantage of the usability of concordancers and the possibility of using the entire Web as a corpus. Moreover, Web Corp enables more complex search options as normal corpora search engines. For example, some search options allow determining the concordance span to full sentences, or filtering the words we want additionally to appear in the documents in which our target word comes up. This is a new strategy we would like to explore before discarding the use of corpora for validating ontology statements.

2. Regarding the use of linguistic patterns, we think that in order to confirm the results from these initial validation tests, we still need to conduct further tests with a higher number of statements. A part from that, we would like to combine some of the techniques exposed here in order to provide reliable results. For example, we could combine the Linguistic pattern instantiation technique, with the WordNet comparison.

3. For further experiments with linguistic patterns, which are the ones that have proved to produce the best results, we would like to refine the patterns we have been using by discarding the ones that have produced no matches at all, and by concentrating on the productive ones. In the case of the LSPs we would like to make some test exploiting the optional elements of this type of patterns, which have been removed in the present test. In the same line of thought, it would be convenient to change the use of the Google exact matching for more flexible ways in which we can automatically identify the two ontology concepts as they are related within sentences, without restricting the exact elements that have to appear and the order they have to follow. This would also allow an enrichment of the LSPs used for this end.

4. In order to validate the results obtained with the Linguistic pattern instantiation technique, we would like to statistically analyze Google results and establish reliable thresholds with the aim of proposing a

[8]http://www.webcorp.org.uk/

valid relation for incorrect ontology statements.

5. We also plan to investigate the use of the WordReference online dictionary that additionally to definitions provides taxonomical trees of terms, in line with WordNet.

6. Finally, we would like to investigate ways of automating the process of instantiation and search in a search engine, corpora, etc.

Chapter 8

Web and Semantic Web based Methods

In this chapter we explore the use of the Web and the Semantic Web as sources of background knowledge for judging the correctness of a given ontology statement. While the first two sections of this chapter are rather experimental in nature, we conclude with a section that provides a formal framework for using online ontologies. This framework does not only cover the use of online ontologies for judging the correctness of a statement, but extends its coverage to the general topic of measuring agreement (or disagreement) between these online ontologies.

8.1 Web-based Measures

Web data has often been used to verify or compute relatedness. For example, Calibrasi et al.'s Google similarity distance is used to measure the similarity between two terms [CV07]. Or, the Pankow system combines linguistic patterns and information provided by Google to verify whether a relationship of a certain type holds between two terms [CLS05]. In this section, we report on how methods derived from these two basic techniques were used for evaluating relations between ontology terms.

In [GM08], Gracia and Mena, present a technique for computing relatedness between ontology concepts (drawn from different ontologies). This technique relies on applying a web-based semantic relatedness (derived with the NGD formula) to the semantic neighborhood of the two concepts. If the components of this neighborhood (i.e., synonyms, hypernym, hyponyms) are sufficiently related to each other, then the concepts are considered related as well.

The same paper has shown that this technique can successfully be used to judge the correctness of mappings between ontology concepts. The hypothesis is that concepts which are very unrelated are unlikely to have a relation between them, so the chance is that the given mapping is false. For example, experiments with our AGROVOC/NALT dataset, have shown that for a given relatedness threshold of 0.19, this method could correctly predict the human judgement for 79% from a set of mappings. Note that the technique was most effective in identifying mappings which were false because of improper anchoring.

For this deliverable, we have extended the experiments for the OAEI'08 datasets as well. We found that the optimal threshold is higher than in the case of the previous data set, namely 0.34. This allowed us to conclude that there is no universally optimal threshold. Its value can change due to the fact that the indexes of the underlying search engines grow and evolve. The value is also domain dependent. Indeed, for the OAEI domain where highly similar ontologies are matched the threshold is necessarily higher than in the AGROVOC/NALT case which covered a variety of very different domains. The precision of the assessment obtained with this threshold was 66%, thus lower than the 79% reported in [GM08]. Our hypothesis in explaining this difference relates again to the high similarity and restricted domain of the evaluated ontologies (in comparison to the much more diverse content of the Agrovoc and Nalt ontologies), which leads to a higher number of bad inferred mappings between highly related terms, thus hampering the optimal functioning of our technique.

For all the relations of the four datasets we have computed the Average and the Median for the various relation types:

Measure	True	False- All	False	F-Anchor	F-PartOf	F-Generic
Average	0.452	0.398	0.407	0.323	0.432	0.396
Median	0.347	0.348	0.352	0.300	0.387	0.362

Table 8.1: Average and Median of the Web-relatedness measure for various relation types in the OAEI'08 datasets.

The values shown in Table 8.1 indicate that, while there is a difference between the average values for true and false statements, this is not very significant (first two columns). A plausible explanation for this behavior is that, unlike the ontologies of the AGROVOC/NALT dataset, the ontologies in this dataset are much more similar (all belonging to the same domain of bibliography) thus leaving little scope for this measure to be effective.

It is interesting to investigate the behavior of this method for various types of false relations. The lowest values are obtained for errors that were introduced by anchoring (i.e., the algorithm that derived the relation has interpreted the sense of one or both of the source concepts incorrectly), thus being consistent with the findings of [GM08]. On the contrary, mappings which are false because they actually represent part-of relations rather than is-a, obtained much higher values with this measure. This is logical since terms that are part of each other are highly related and should be picked up as such by this measure.

Based on these experiments we conclude that this web-based measure performs well for predicting the correctness of a relation established between two concepts (not just terms). Our intuition is that this method could be further explored by combining it with other measures.

8.2 Semantic Web based Methods

The previous section and Chapter 7 have presented a set of methods where the Web is used as a source of background knowledge to solve various tasks such as computing the relatedness between two terms or the frequency of the appearance of certain linguistic patterns. Indeed, there is a large body of work in exploring the Web as a language resource.

In this section, we focus on the use of the Semantic Web (i.e., online available semantic data and search engines for exploring these) to identify methods for estimating the correctness of a given statement. Our exploration is inspired by similar work in language technology where two core types of language resources (LR) are explored. On the one hand, large-scale, un- or weakly-structured resources (e.g., corpora, the textual Web) are explored through statistics based methods to derive useful information. On the other hand, manually crafted semantic resources (e.g., dictionaries or thesauri such as WordNet) are explored through semantic methods. Interestingly, the SW exhibits the characteristics of both these LR types being a large-scale collection of structured resources (i.e., ontologies).

Our hypothesis is that methods developed in the domain of language technology can be adapted to the Semantic Web. Therefore, we have experimented with a set of such methods. Since this work is the first one in exploring the SW as a language resource, we give a full account of our explorations including even those methods that did not lead to particularly good results.

8.2.1 Methods using Path Heuristics

In their extensive survey of methods for measuring semantic distance using WordNet [BH06], Budanitsky and Hirst state that most methods exploring lexical resources exploit the notion of path in these resources. Literally:

"All approaches to measuring semantic relatedness that use a lexical resource construe the resource, in one way or another, as a network or directed graph, and then base the measure of relatedness on properties of paths in this graph."

For example, a simple measure relies on the observation that the length of a path between two terms in a lexical resources correlates with their relatedness (i.e., the closer the terms are in the network, the more related they are). Although this is one of the simplest measures it has nevertheless lead to one of the best results [BH06].

The Semantic Web is a collection of ontologies and thus it permits the use of such path based methods. Indeed, let $< s, R, t >$ be a relation which we wish to evaluate, where R is the name of the relation, s is its source (or domain) and t is its target (or range). We also suppose that there are a set of n online ontologies such that each ontology O_i contains concepts similar to s and t ($s = s'_i$ and $t = t'_i$) and the path between these (P_i) leads to the derivation of a relation equivalent to R ($R = R'_i$). Adapting the results of language technology, we consider that the length of P_i is directly proportional with the relatedness between s and t. We extend this with the hypothesis that the relatedness degree between s and t should give an indication about the correctness of the relation that is established between them.

In order to find out the set of ontologies with the characteristics mentioned above, we can use the Scarlet relation discovery service. Indeed, the first strategy of this algorithm identifies all ontologies that contain a path between s and t and returns this path. We can then filter out those ontologies that lead to the derivation of R. Our own experience while evaluating relations derived by Scarlet in various different contexts is that there are some characteristics of the derivation path that correlate with the correctness of the statement. In particular we have established the following characteristics:

The length of the path As discussed above, this is a core measure adapted from language technology research.

Content of path Paths that contain abstract concepts e.g. (thing, root, agent, event, collection, list, individual) incorporated due to the transitivity of inheritance or less likely to be correct.

Type of relation The type of the relation might also have an influence on its potential correctness. Our hypothesis is that some types of relations might be easier to extract correctly than others.

To measure correctness of a given relation we compute the mean length of the paths that lead to the derivation of this relation:

$$MeanPathLength_R = \frac{\sum_i Length(P_i)}{n}$$

Evaluation: Table 8.2 shows the results of our experiment aimed at understanding the correlation of the path length with the correctness of a relation. To that end, we computed $MeanPathLength_R$ for all relations in five of our datasets and then we computed the mean pathlength for relations judged to be false and those judged to be true. The results suggest that the path length value clearly differ entities between true and false relations. Indeed, most true relations have mean paths that are less than 2, while false relations have path lengths higher than 2.5. A valuable conclusion is that in the case of parameterized algorithms such as Scarlet, inheritance depth can be restricted when precision is of high importance.

In Table 8.3 we show the ratio of correct relations for four different relation types. According to these figures, disjointness relations are the easiest to predict and they are almost always correct. Generally, subclass and superclass relations can also be detected with a good precision, while generic relations often introduce the most errors (with the notable exception of the last two data sets). While not discriminative enough to be used on its own, this simple measure could be combined with other methods.

Data Set	F-Mean	T-Mean
OAEI'08 301	2.585	1.167
OAEI'08 302	2.562	1.363
OAEI'08 303	2.734	1.625
OAEI'08 304	2.428	1.779
AKT	2.125	2.411

Table 8.2: Correlation between path length and the correctness of a statement.

Data Set	Disjoint	SubClass	SuperClass	Generic
OAEI'08 301	100%	98%	86%	37%
OAEI'08 302	100%	86%	65%	30%
OAEI'08 303	100%	70%	70%	43%
OAEI'08 304	100%	55%	74%	65%
AKT	100%	86%	40%	76%

Table 8.3: Correlation between relation types and precision.

8.2.2 Co-ocurence based methods

Another big family of methods used to derive relatedness consists of distributional measures, i.e., measures that use statistic techniques to assess how similar the context of two words are. We have experimented with two such measures.

First, the Jaccard coefficient measures the probability of two words appearing together through the ratio of contexts in which the words appear together over all context where the words appear (together or separately). We adopted this measure to the Semantic Web by measuring the ratio of ontologies which mention both concepts together (s and t) and of ontologies that mention at least one of the concepts. The rationale behind this measure is that two terms are related if they often appear to be mentioned in the same document. Formally:

$$M = \frac{|O_s \bigcap O_t|}{|O_s \bigcup O_t|}$$

Second, we have adapted the Normalized Google Distance [CV07] to Watson, and called it NWD - Normalized Watson distance. $f(x)$ denotes the number of ontologies where x appears as a concept.

$$NWD_{s,t} = \frac{\max\{\log f(s), \log f(t)\} - \log f(s,t)}{\log N - \min\{\log f(s), \log f(t)\}}$$

Since these measures estimate relatedness rather than the correctness of a relation, we have evaluated them on a set of benchmarks for semantic relatedness estimation. The correlation of the Jaccard value with the largest available benchmark, WordSim353[1], was very low, only 0.14. The comparison of the NWD with this benchmark was even lower and equaled 0.10. However, this benchmark has several shortcomings, for example that the assignments of correlation is not symmetric (e.g., the correlation value of the pair of (bank, money) is different than the one for (money, bank)). Therefore, we have compared with a novel dataset built by J. Gracia that addresses the limitations of the previous benchmark [GM08] and obtained the value of 0.48. This is quite a good value given that the value of maximum correlation is 1 and that four out of seven WordNet relatedness measures performed worse on this dataset [GM08]. Nevertheless, we can conclude that such statistic measures do not function at the current scale and coverage of the Semantic Web.

[1]http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/wordsim353.html

8.2.3 Semantic Agreement

Our final measure derives a correctness value for a certain statement based on the level of agreement of online ontologies. As such it is different from the two previous sets of methods which were inspired from NLP techniques. Given a statement $< s, r, t >$ our first metric measures the ratio between the ontologies from which r can be derived between s and t and all ontologies from which any relation between these terms can be derived. Intuitively, this measure quantifies the importance of the relation over all relations that can be identified between s and t. Note, however, that other variations of this metric can also be implemented, e.g., reporting the number of ontologies leading to r to all ontologies that mention s and t. Formally:

$$A_r = \frac{|O_r|}{|O_R|}$$

This measure has the property that:

$$\sum_i A_{r_i} = 1$$

An interesting idea is to investigate those cases where there is a conflicting situation in the relation set when $|R| > 1$. Since the semantics of named relations is not formally defined, we cannot include them in the calculation of conflicts. For the remaining relations, namely $\subseteq, \supseteq, \perp, sibling$, we consider:

- a weak conflict when $\subseteq, \supseteq$ appear together;
- a strong conflict when either $\subseteq$ or $\supseteq$ appear together with $\perp$.
- a strong conflict when either $\subseteq$ or $\supseteq$ appear together with $sibling$.

Contrary to conflicts, some pairs of relations can actually re-enforce each others. For example, when the $sibling$ relation is re-inforced by $\perp$ (siblings are often disjoint), and vice-versa.

Evaluation: Table 8.4 presents the average semantic agreement obtained for false (F) and true (T) relations. The results suggest that this measure has some correlation with correctness value: indeed, in all except the last dataset, the average value for false relations is always lower than that of true relations. At the same time, we observe that it is hard to predict a global threshold as mean values for true and false statements are not very consistent over the five datasets.

Data Set	F-Mean	T-Mean
OAEI'08 301	0.0025	0.3413
OAEI'08 302	0.127	0.315
OAEI'08 303	0.127	0.161
OAEI'08 304	0.175	0.266
AKT	0.287	0.262

Table 8.4: Overview of SAM results.

We have also found no significant correlation between conflicts and correctness value. We noted, however, that most disjoint relations are endorsed by an endorsement (i.e., they usually co-occur with a $sibling$ relation).

We therefore conclude that semantic agreement measures are promising and should be further investigated. Indeed, in what follows we present an initial formal framework for leveraging semantic agreement on the Semantic Web.

8.2.4 Implementation Details

All the measures described in this section have been implemented in a dedicated java library. This library has been primarily used for testing purposes but we plan to extend it and release it so that it can be integrated in other tools. In particular, the Evolva plugin will provide the first test-bed for including these measures. Their values will be used to rank the automatically learned relations so that the users are better supported in their decision to chose a relation or not.

8.2.5 Conclusions and Future Work

In this section we have presented a suite of novel methods that exploit the Semantic Web as a source of knowledge for judging the correctness of a given statement. From our experimental investigations we conclude that Path based measures and measures exploring semantic agreement are the most promising and should be further investigated. Indeed, in the next section we take our first step in this direction by presenting a formal framework for measuring agreement between online ontologies. As future work, we plan to extend this framework and validate it with appropriate experiments.

8.3 Formally Measuring Agreement and Disagreement in Ontologies

Ontologies are knowledge artifacts representing particular models of some particular domains. They are supposed to be shared within the communities that rely on them, meaning that they represent consensual representations inside these communities. However, several ontologies can cover the same domain, while being built by and for different communities. In these cases, different perspectives, points of view and opinions might be expressed on the same concepts and objects of the domain, i.e. ontologies might disagree.

Knowing to which extent an ontology agrees with another one, or with a particular statement, can be very useful in many scenarios. When using several ontologies in a common application, for example, disagreement can create unexpected results for the user. Also, when reusing statements from other ontologies while building a new one (e.g., using the Watson Plugin [dSM08]), there is a need for some formal measures to indicate whether or not there exists a common agreement concerning a candidate statement, in the current ontology or in other, external ontologies. Realizing that there is a high level of disagreement in available ontologies with a given statement/ontology, or that there is no clear cut between agreement and disagreement can then be very useful in pointing out potentially problematic statements/ontologies. Finally, when matching ontologies based on background knowledge (e.g., with Scarlet [SdM08b]), one useful criterion for selecting an ontology as background knowledge is the level of agreement it has with the matched ontologies. An initial measure of agreement has been devised and tested for this purpose in Section 8.2.3. We consider here a more formal and generic approach to measuring agreement.

One way to assess whether two ontologies disagree is to rely on the presence or absence of logical contradictions. The two ontologies can be merged, based on mappings between their entities, and the resulting model be checked for inconsistency and incoherence (the same approach can be used to check if an ontology disagrees with a statement). While this approach would certainly detect some forms of disagreement, it suffers from a number of limitations. First, it only checks whether the ontologies disagree or not. It does not provide any granular notion of disagreement and, if no contradictions are detected, it does not necessarily mean that the ontologies agree. Indeed, while two ontologies about two completely different, non overlapping domains would certainly not disagree, they do not agree either. More importantly, logical contradictions are not the only way for two ontologies to disagree. Indeed, there could also be conceptual mismatches, like in the case where one ontology declares that “Lion is a subclass of Species” and the other one indicates that “Lion is an instance of Species”. Even at content level, logical contradictions would not detect some form of disagreements. Indeed, the two statements “Human is a subclass of Animal” and “Animal is a subclass of Human” do not generate any incoherence. However, they disagree in the sense that, if put together, they generate results that were not expected from any of the two ontologies.

In this section, we describe a set of measures to assess the levels of agreement and disagreement between an ontology and either a statement, or a full ontology. These measures are designed to take into account the formal semantics of the considered ontologies, not only to detect logical contradictions, but also other kinds of contradictions that would not be covered by purely logical approaches.

8.3.1 Assumptions and Requirements

Based on the usage scenarios quickly sketched above and on the analysis of the limitations of the basic approach based on checking logical contradictions, we devised a number of requirements that our measures should fulfil:

R1: Ontologies agree with themselves. One of the starting points is that we need to consider that an ontology agrees with itself, meaning that our measures should provide maximal results for agreement (and no disagreement) when checking an ontology against itself. It also means that we assume the ontologies we assess to be consistent, coherent, and homogeneous in terms of modeling.

R2: Covering different domains is not agreeing. We consider that there is a fundamental difference between agreeing and not talking about the same things. This means that, when applied on ontologies or statements that cover completely different domains (i.e., for which there is no mapping between their entities), our measures should indicate that there is no disagreement and no agreement.

R3: Different levels of agreement/disagreement. As already mentioned, there are different reasons for which an ontology might disagree with another one (at the level of the content, logically, at the level of the representation, etc.) These different kinds of disagreements should be considered with different levels of importance, leading to granular measures rather than binary tests.

R4: Independent from matching techniques. In order to check the agreement/disagreement of an ontology with respect to another ontology or a statement, corresponding entities should be related through mappings. However, the way these mappings are produced should not affect the computation of the agreement/disagreement measures. Therefore, we will assume the existence of mappings we can exploit, without considering any particular approach to produce such mappings. In our experiments, we will use a simple technique based on the lexical equivalence of the entities' names.

R5: It is possible to agree and disagree at the same time. Two ontologies may agree on some statements and disagree on some others, meaning that, depending on the number of statements in each category and on the level of agreement/disagreement for each statement, the two ontologies might agree and disagree at different levels. It should even be possible for an ontology to both agree and disagree with one single statement. Indeed, we consider the case of an ontology that contains the statement "Lion is a subclass of Species" and for which we want to check whether it agrees with the statement "Lion is an instance of Species". As already mentioned, this ontology disagree at modeling level with this statement. However, at content level, it agrees to a certain extent that Lion is included in Species.

8.3.2 Basic Framework

In this section, we consider an ontology O to be defined as a set of statements (corresponding to axioms in the description logic terminology). A statement is the expression of a relation between two entities of the ontology in the form of a triple $< subject, relation, object >$. In RDF-based ontology languages, $relation$ corresponds to the identifier of a property (its URI), $subject$ corresponds either to the URI of an entity or to an anonymous resource, and $object$ corresponds to the URI of an entity, an anonymous resource or a literal. In the following, we base our definition of agreement and disagreement on whether or not an ontology conflicts with the relation linking two named entities. Hence, we chose to ignore the cases of anonymous entities (i.e., blank nodes in RDF) and literals. This does not mean that our measures could not be applied to ontologies containing such elements, but simply that we will only use statements where the subjects and objects are identifiers of entities.

As already mentioned, the atomic element on which ontologies can express agreement or disagreement is the statement. Indeed, an ontology can contain information that contradict, or on the contrary enforce the one expressed through a particular statement, and so can disagree or agree with the relation expressed by this statement. We then define to elementary functions representing respectively the agreement and the disagreement levels of an ontology O with respect to a statement $s =< subject, relation, object >$:

$$agreement(O, s) \rightarrow [0..1]$$
$$disagreement(O, s) \rightarrow [0..1]$$

We chose to use two distinct measures for agreement and disagreement so that an ontology can, at the same time and to certain extents, agree and disagree with a statement (cf. R3, R5). These two measures have to be interpreted together to indicate the particular belief expressed by the ontology O with regards to the statement s. For example, if $agreement(O, s) = 1$ and $disagreement(O, s) = 0$, it means that O fully agrees with s and conversely if $agreement(O, s) = 0$ and $disagreement(O, s) = 1$, it fully disagrees with s. Now, agreement and disagreement can vary between 0 and 1, meaning that O can only partially agree or disagree with s (cf. R3) and sometime both, when $agreement(O, s) > 0$ and $disagreement(O, s) > 0$ (cf. R5). Finally, another case is when $agreement(O, s) = 0$ and $disagreement(O, s) = 0$. This basically means that O neither agrees nor disagrees with s, for the reason that it does not express any belief regarding the relation encoded by s (cf. R2).

The actual values returned for both measures, when different from 0 and 1, are not very important, as they should only correspond to different levels of dis/agreement, meaning that we only need to define an order between pre-defined levels, without the need for fixing the actual values for these levels. In this work, we consider two different levels of agreement different from 0 and 1, defined as the following:

$$0 < A1 < A2 < 1$$

We also introduce 3 levels of disagreement:

$$0 < D2 < D1 < 1$$

Note again that the actual values for these levels are not significant, only the total order between them. However, to facilitate comparisons of measures, desirable properties are that $A1 + A2 = 1$, $D1 + D2 = 1$, $A1 + D1 = 1$ and $A2 + D2 = 1$.

To be able to compute a (non-null) level of dis/agreement, it is necessary that the ontology O covers the entities related by the statement s. Therefore, we need to identify entities in O that correspond to the entities in the tested statement s. For this, we assume the existence of a function $match(O, e)$ that returns the identifier of an entity e' in O (cf. R4). To simplify the notation, we will call s' and o' the matching entities in an ontology O of the subject and object of a statement $< s, r, o >$ respectively.

Note that the $match$ function should not be completely defined, meaning that it can happen that no matching entity is found in O for s or o. However, the following definitions will assume the existence of s' and o', hence we consider that $agreement(O, t) = 0$ and $disagreement(O, t) = 0$, with $t =< s, r, o >$, if and only if $match(O, s)$ or $match(O, o)$ are undefined.

8.3.3 Measuring Agreement and Disagreement of an Ontology Regarding a Statement

Based on the basic definitions of the previous section, we can now devise a formal way to compute the agreement and disagreement measures between statements and ontologies. As mentioned before, these measures should essentially assess whether or not the relation between entities expressed in the considered statement s is 'validated' by the ontology. The central element to consider for our measures is therefore the type of the relation r in the considered statement $< s, r, o >$. This relation can either be a property included in the ontology representation language, i.e., one of $\{subClassOf, equivalentClass, domain, range, disjointWith, type, sameAs, differentFrom, subPropertyOf\}$, or a generic, user-defined property R.

To assess (dis)agreement between this statement and the ontology O, we then need to compare the relation encoded in the statement, and the relations that are expressed between corresponding entities in O. In other terms, we need to extract from O statements that express relations between s' and o', the entities matching s and o. Note that these statements should not be necessarily the ones declared in O, but any statement that O *entails* and that relates s' and o'. This leads to the definition of the *R-module* of an ontology O with respect to a statement $< s, r, o >$.

The **R-module** of an ontology O with respect to a statement $< s, r, o >$ is the list of statements of the form $< s', ?, o' >$ or $< o', ?, s' >$ that are entailed by O. Formally, it is defined by $RM(O, < s, r, o >) = \{st =< s', r', o' > \ or \ st =< o', r', s' > \mid match(O, s) = s' \ \wedge \ match(O, o) = o' \ \wedge \ O \models st\}$. Note that RM can be empty, meaning that the ontology does not express any relation between the two given entities.

The R-module represents a complete set of statements that O directly or indirectly expresses about the relations between s and o. In order to make it a minimal, concise summary of these relations, we introduce the notion of *non-redundant R-module*.

The **non-redudant R-module** of an ontology O with respect to a statement $< s, r, o >$ is a subset of the corresponding R-module such that none of the statements included can be inferred from other statements in the set. Formally, it is defined by $MRM(O, < s, r, o >) = \{st =< s', r', o' > \ or \ st =< o', r', s' > \mid st \in RM(O, < s, r, o >) \ \wedge \ RM(O, < s, r, o >) \setminus \{st\} \not\models st\}$.

In addition, according to requirement R1, we assume that:

1. $RM(O, < s, r, o >)$ is consistent and coherent set of axioms, meaning for example that it cannot contain a statement $< s, subClassOf, o >$ and a statement $< s, disjointWith, o >$.

2. $RM(O, < s, r, o >)$ does not contain any modeling conflict, meaning for example that it should not imply that an entity is at the same time a class and an individual, like in the two statements $< s, subClassOf, o >$ and $< s, type, o >$.

As for the relation in the considered statement, the MRM of an ontology can contain any kind of relation, in particular properties of the language and user defined relations. In the later case, we need to distinguish between the cases of relations that match the one described in the statement (i.e., when $match(O, r) = r'$) and the cases of relations that do not match.

In order to simplify the notations, we represent the set of statements in a MRM as a set of relations between s' and o'. In this notation, a statement such as $< s', subClassOf, o' >$ is represented by the relation $subClassOf$ and the statement $< o', subClassOf, s' >$ by the relation $subClassOf^{-1}$. Concerning user-defined properties, a statement $< s', r', o' >$ is represented by $matchRelation$ if $match(O, r) = r'$, r being the relation in the considered statement, and by $mismatchRelation$ if not.

One of the interesting properties of the set MRM in this notation is that, due to the fact that it is non-redundant, consistent and coherent, and that it does not contain conflicts at modeling level, there exists a finite set of possible MRM whatever are the considered ontology and the considered statement. In other terms, there exist only a restricted number of allowed combinations of relations, as MRM sets such as $\{subClassOf, subClassOf^{-1}, equivalentClass\}$, $\{subClassOf, disjointWith\}$ or $\{subClassOf, type\}$ do not comply with the definition.

Assessing the level of agreement and disagreement between a statement and an ontology consists in comparing the relation r expressed in the statement, and the corresponding MRM from O, to check whether they enforce or contradict each other. Considering that there are only a finite set of possible situations, the result of the agreement (resp. disagreement) measure can be completely defined by a matrix establishing the returned values depending on the relation r and on the set MRM. The matrix we propose for the agreement measure is shown in Table 8.5, and the one for disagreement is shown in Table 8.6.

As an example, if we consider the statement $st =< Person, sameAs, Human >$ and as MRM for the ontology O the set $\{subClassOf\}$, by inspecting Table 8.5 and Table 8.6, we obtain the following values for the measures of agreement and disagreement:

- $agreement(st, O) = A1$
- $disagreement(st, O) = D1$

indicating that st and O both partially agree (because both believe there is an overlap between $Human$ and $Person$) and partially disagree (because they do not express the same level of overlap, and they use different modeling, considering classes in one case and instances in the other case).

8.3.4 Measuring Agreement and Disagreement between two Ontologies

Considering that ontologies are made of statements, extending the measures above to compute agreement and disagreement between two ontologies should be relatively straightforward by computing the average of each measure for each statement of an ontology against the other ontology. However, there are two issues related to this approach. First, the measures would then not be symmetric, as declared statements would be considered in one case, and entailed ones in the other case. Second, in cases where the ontologies only have a small overlap, the results would be lowered down by all the statements which would not find matches from one ontology to the other (resulting in null values for both agreement and disagreement).

For these reasons, we define the measures of agreement and disagreement between ontologies in the following way:

- $agreement(O_1, O_2) = \frac{\sum_{st \in ST_1} agreement(st,O_2) + \sum_{st \in ST_2} agreement(st,O_1)}{|ST_1|+|ST_2|}$
- $disagreement(O_1, O_2) = \frac{\sum_{st \in ST_1} disagreement(st,O_2) + \sum_{st \in ST_2} disagreement(st,O_1)}{|ST_1|+|ST_2|}$

where $ST_1 = \{< s, p, o > \in O_1 \mid match(s, O_2) \neq \emptyset \wedge match(o, O_2) \neq \emptyset\}$ and $ST_2 = \{< s, p, o > \in O_2 \mid match(s, O_1) \neq \emptyset \wedge match(o, O_1) \neq \emptyset\}$. In this way, only relevant statements are checked for agreement and disagreement (ST_1 and ST_2) and the measures are actually symmetric.

8.3.5 Measuring Consensus and Controversies

In some scenarios, like the one of the Watson plugin, it is also useful to consider global measures computed on an entire ontology repository, to assess on how much *a statement is (dis)agreed* in available ontologies, i.e., if there is a consensus on whether this statement is correct or not. Conversely, a related information concerns the level of controversy on the statement, i.e. whether there is a clear cut between agreement and disagreement.

To compute such measures, we first need a measure of the global agreement between the considered ontology repository and a statement st. We represent the ontology repository as a set R of ontologies. In our experiments this set corresponds to Watson's collection of ontologies. We then compute the agreement and disagreement between a statement st and this repository R as a simple average, but taking only into account ontologies matching the entities of st:

- $agreement(st, R) = \frac{\sum_{O \in rR} agreement(st,O)}{|rR|}$
- $disagreement(st, R) = \frac{\sum_{O \in rR} disagreement(st,O)}{|rR|}$

where $st = < s, p, o >$ and $rR = \{O \in R \mid match(s, O) \neq \emptyset \wedge match(o, O) \neq \emptyset\}$.

The level of consensus concerning a statement st corresponds to the level certainty on either the global agreement or the global disagreement within the ontology repository R. We say that there is a high level of (positive) consensus if the overall agreement about this statement is high, and the overall disagreement is low. Thus, we define the measure of consensus in a set of ontologies R upon a statement st as follows:

r	sClass	equiv.	dom.	range	disj.	type	same	diff.	sProp.	R
$\emptyset$	0	0	0	0	0	0	0	0	0	0
sClass	1	A2	0	0	0	A2	A1	0	A1	0
sClass^{-1}	A1	A2	0	0	0	0	A1	0	0	0
equiv.	A2	1	0	0	0	A1	A2	0	A1	0
disj.	0	0	0	0	1	0	0	A2	0	0
dom.	0	0	1	0	0	0	0	0	0	0
dom.$^{-1}$	0	0	0	0	0	0	0	0	0	0
range	0	0	0	1	0	0	0	0	0	0
range^{-1}	0	0	0	0	0	0	0	0	0	0
dom., range	0	0	1	1	0	0	0	0	0	0
dom.$^{-1}$, range^{-1}	0	0	0	0	0	0	0	0	0	0
sProp	A2	A1	0	0	0	A1	A1	0	1	0
sProp^{-1}	0	A1	0	0	0	0	A1	0	A1	0
type	A2	A1	0	0	0	1	A1	0	A1	0
type^{-1}	0	A1	0	0	0	A1	A1	0	0	0
same	A1	A2	0	0	0	A1	1	0	A1	0
match	0	0	0	0	0	0	0	0	0	1
match^{-1}	0	0	0	0	0	0	0	0	0	0
mmatch	0	0	0	0	0	0	0	0	0	0
same, match	A1	A2	0	0	0	A1	1	0	A1	1
same, match^{-1}	A1	A2	0	0	0	A1	1	0	A1	0
same, mmatch	A1	A2	0	0	0	A1	1	0	A1	0
diff.	0	0	0	0	A2	0	0	1	0	0
diff. match	0	0	0	0	A2	0	0	1	0	1
diff., match^{-1}	0	0	0	0	A2	0	0	1	0	0
diff., mmatch	0	0	0	0	A2	0	0	1	0	0

Table 8.5: Matrix definition of the agreement measure, depending on the relation r in the considered statement, and the set of relations in the MRM from the considered ontology O. R represents a user-defined property. Other relations are abbreviated.

r	sClass	equiv.	dom.	range	disj.	type	same	diff.	sProp.	R
$\emptyset$	D1	D1	D2	D2	D1	D1	D1	D1	D1	D2
sClass	0	D2	D1	D1	1	D2	D1	1	D2	D1
sClass^{-1}	D1	D2	D1	D1	1	D1	D2	1	D1	D1
equiv.	D2	0	D1	D1	1	D1	D2	1	D1	D1
disj.	1	1	D1	D1	0	1	1	D2	1	D1
dom.	D1	D1	0	0	D1	D1	D1	D1	D1	D1
dom.$^{-1}$	D1	D1	D1	D1	D1	D1	D1	D1	D1	D1
range	D1	D1	0	0	D1	D1	D1	D1	D1	D1
range^{-1}	D1	D1	D1	D1	D1	D1	D1	D1	D1	D1
dom. range	D1	D1	0	0	D1	D1	D1	D1	D1	D1
dom.$^{-1}$ range^{-1}	D1	D1	D1	D1	D1	D1	D1	D1	D1	D1
sProp	D2	D1	D1	D1	1	D1	D1	1	0	D1
sProp^{-1}	D1	D1	D1	D1	1	D1	D1	1	D1	D1
type	D2	D1	D1	D1	1	0	D1	1	D1	D1
type^{-1}	D1	D1	D1	D1	1	D1	D1	1	D1	D1
same	D1	D2	D1	D1	1	D1	0	1	D1	D1
match	D1	D1	D1	D1	D1	D1	0	0	D1	0
match^{-1}	D1	D1	D1	D1	D1	D1	0	0	D1	D1
mmatch	D1	D1	D1	D1	D1	D1	0	0	D1	D2
same match	D1	D2	D1	D1	1	D1	0	1	D1	0
same match^{-1}	D1	D2	D1	D1	1	D1	0	1	D1	D1
same mmatch	D1	D2	D1	D1	1	D1	0	1	D1	D2
diff.	1	1	D1	D1	D2	1	1	0	1	D2
diff. match	1	1	D1	D1	D2	1	1	0	1	0
diff. match^{-1}	1	1	D1	D1	D2	1	1	0	1	D1
diff. mmatch	1	1	D1	D1	D2	1	1	0	1	D2

Table 8.6: Matrix definition of the disagreement measure, depending on the relation r in the considered statement, and the set of relations in the MRM from the considered ontology O. R represents a user-defined property. Other relations are abbreviated.

Statement	Nb1	Nb2	a	d	cs	ct
$< SeaFood, disjointWith, Dessert >$	12	12	1.0	0.0	1.0	0.0
$< Fowl, disjointWith, SeaFood >$	11	11	1.0	0.0	1.0	0.0
$< Pasta, disjointWith, SeaFood >$	12	12	1.0	0.0	1.0	0.0
$< SeaFood, subClassOf, EdibleThing >$	11	11	1.0	0.0	1.0	0.0
$< ShellFish, subClassOf, SeaFood >$	14	16	0.937	0.047	0.89	0.109
$< Fish, subClassOf, SeaFood >$	12	14	0.928	0.053	0.875	0.125
$< SeaFood, disjointWith, Fruit >$	11	14	0.85	0.1	0.75	0.25
$< Meat, disjointWith, SeaFood >$	8	16	0.75	0.22	0.53	0.46
$< SeaFood, subClassOf, Meat >$	4	16	0.125	0.844	-0.719	0.281

Table 8.7: Results from applying agreement and disagreement measures on statements obtained from the Watson Plugin with SeaFood. Nb1 is the number of ontologies in which the statement appear. Nb2 is the number of ontologies containing entities matching the subject and object of the statement. 'a' (resp. 'd') is the level of agreement (resp. disagreement) with the statement in Watson. cs (resp. ct) is the level of consensus (resp. controversy) on the statement in Watson.

$$consensus(st, R) = agreement(st, R) - disagreement(st, R)$$

this measure should be interpreted in the following way.

- if $consensus(st, R) > 0$ then it represents the level of 'positive' consensus in R about st, meaning the consensus on agreement.
- if $consensus(st, R) < 0$ then it represents the level of 'negative' consensus in R about st, meaning the consensus on disagreement.

We consider the notion of controversy to be the inverse from the one of consensus: there is a high level of controversy on a given statement when there is no clear cut between agreement and disagreement, i.e. there is a low level of consensus. Therefore, the measure of controversy in a set of ontologies R upon a statement st can simply be computed in the following way:

$$contreversy(st, R) = 1 - |agreement(st, R)|$$

8.3.6 Experiment

To test the behavior of our measures, we experiment on computing them on a concrete scenario, using the Watson Plugin. We consider the case of an ontology engineer who needs to integrate to her ontology the class *SeaFood*. The Watson Plugin retrieves a set of statements for this class, from other ontologies. We then compute the overall agreement and disagreement measures in Watson for each of these statements, as well as the consensus and controversy measures.

For this experiment, we use a simple, lexical matching technique, and define $A1 = 0.25$, $A2 = 0.75$, $D2 = 0.25$ and $D1 = 0.25$. The results are summarized in Table 8.7. Note that, in this table, 'lexically similar' statements are grouped together (the table indicates the number of ontologies where each statement appear).

As can be seen from these results, the 4 first statements are fully agreed with by ontologies in Watson, meaning that all the ontologies containing both entities of each statements express exactly the same relation as the one of the statement. The 3 next statements also have a very high level of agreement, and a very low level of disagreement. This is mainly due to a few ontologies containing the right entities, but not necessarily relating them. Finally the 2 last statements are the ones for which there is the highest level of controversy. The last one is by far the most disagreed with (which correlates with the high level of agreement of the other one, which contradicts it).

Another interesting example is the one of the statement $< river, subClassOf, sea >$, which gives a high level of disagreement (0.766). The disagreement is not 1 in that case, because only very few ontologies express explicitly contradicting relations. However, in this case, the level of agreement is 0: There is no ontology to actually agree with this statement.

8.3.7 Related Work

Belief revision (see for example [AGM85]) is concerned with the consistent update and modification of a knowledge bases, using ‘revision operators’ to ensure that the result of the modification is a coherent model, conserving as much as possible from the original knowledge base. It is related to this work in the sense that it considers a knowledge base (an ontology) as a belief that can be challenged. In our case however, it is not challenged by new knowledge, but by other ontologies. Also, while belief revision provides ways to solve conflicts in revision, our goal is to assess these conflicts at various levels.

As already mentioned, a common way to check whether two ontologies are ‘logically compatible’ is to merge the ontologies, and to check for inconsistencies and incoherences. However, this only check for one particular type of disagreement. Another type of disagreement could be checked by using the notion of *conservative extension* [GLW06]. Informally, a conservative extension of an ontology is an ontology where axioms have been added which do not ‘affect’ the relations that can be inferred about the entities of the original ontologies. Merging two ontologies and checking whether the result is a conservative extension of both original ontology would give indications of some forms of disagreement. However, this would only give a ‘black box test’, not providing a gradual notion of disagreement, neither a proper notion of agreement.

8.3.8 Future Work

We have realized a straightforward implementation of all the measures described above and tested them on simple examples. At a short term, we intend to provide these measures as Web services on top of Watson, allowing external applications to check agreement related measures for particular statements. In particular, these services will be used in the Watson plugin to provide useful indications to help the user in selecting good statements, and to rank them according to their level of agreement/disagreement on the Semantic Web. Also, the measures can be used directly in the Watson plugin to check whether proposed statements agree or disagree with the ontology being developed.

Another interesting element to explore would be to compute ‘explanations’ for the measures, showing what statements conflict or enforce the one which is tested and, in addition to providing measures of agreements and disagreements between ontologies, to also compute precisely ‘on what’ the considered ontologies agree and disagree.

Appendix A

Detailed LSP Specifications

LSP Identifier	An acronym composed of LSP + ODP component + ISO code for language
NeOn ODPs Identifier	An acronym composed of component type + component + number
Formalization	LSPs formalized according to BNF extension
Examples	Sentences in NL that exemplify corresponding LSPs

Table A.1: Template for the description of LSPs.

LSP Identifier		LSP-SC-EN
NeOn ODPs Identifier		LP-SC-01
Formalization	1	[(NP<subclass>,)* and] NP<subclass> be [CN] NP<superclass>
	2	[(NP<subclass>,)* and] NP<subclass> (group in|into|as) | (fall into) | (belong to) CN NP<superclass>
	3	There are CD | QUAN [CN] NP<superclass> PARA [(NP<subclass>,)* and] NP<subclass>
	4	[A(n) | QUAN] example | examples | [CN] of NP<superclass> be | include [(NP<subclass>,)* and] NP<subclass>
Examples	1	*An orphan drug is a type of drug.* *Odometry, speedometry and GPS are types of sensors.*
	2	*Thyroid medicines belong to the general group of hormone medicines.*
	3	*There are two types of narcotic analgesics: the opiates and the opioids.* *There are several kinds of memory: fast, expensive, short term memory, and long-term memory.*
	4	*Some examples of peripherals are keyboards, mice, monitors, printers, scanners, disk and tape drives, microphones, speakers, joysticks, plotters and cameras.* *Types of criteria for assessing applications are: quality, safety and efficacy.*

Table A.2: LSPs corresponding to *subclass-of relation* ODP.

LSP Identifier		LSP-SC-Di-EN
NeOn ODPs Identifier		LP-SC-01 + LP-Di-01
Formalization	1	NP<superclass> be | CATV [either] NP<subclass> or NP<subclass>
	2	NP<superclass> be divide | split | separate | group in|into [either] [(NP<class >,)* and] NP<class>
Examples	1	*Animals are either vertebrates or invertebrates.*
	2	*Sensors are divided into two groups: contact and non-contact sensors.*

Table A.3: LSP including *subclass-of* relation and *disjoint* classes ODPs.

LSP Identifier		LSP-SC-PW-EN
NeOn ODPs Identifier		LP-SC-01 CP-PW-01
Formalization	1	NP<class> include | contain | (consist of) | comprise [(NP<class >,)* and] NP<class>
	2	NP<class> be divided | split | separate in|into | [CN] [(NP<class >,)* and] NP<class>
Examples	1	*Arthropods include insects, crustaceans, spiders, scorpions, and centipedes. (LP-SC-01)* *Common mass storage devices include disk drives and tape drives. (LP-SC-01)* *Reproductive structures in female insects include ovaries, bursa copulatrix and uterus. (CP-PW-01)*
	2	*Marine mammals are divided into three orders: Carnivora, Sirenia and Cetacea. (LP-SC-01).* *The cerebrumis divided into two major parts: the right cerebral hemisphere and left cerebral hemisphere. (CP-PW-01)*

Table A.4: LSPs corresponding to *subclass-of relation* and *simple part-whole relation* ODPs.

LSP Identifier		LSP-PW-EN
NeOn ODPs Identifier		CP-PW-01
Formalization	1	(NP<part>,)* and NP<part> COMP [CN] NP<whole>
	2	NP<whole> be COMP [CN] (NP<part>,)* and NP<part>
Examples	1	*Proteins form part of the cell membrane.*
	2	*A state machine workflow is made up of a set of states, transitions, and actions.*

Table A.5: LSPs corresponding to simple *part-whole relation* ODP.

LSP Identifier	LSP-OP-DP-PW-EN	
NeOn ODPs Identifier	LP-OP-01 LP-DP-01 CP-PW-01	
Formalization	1	NP<class> have NP<class>
Examples	1	*Birds have feathers.* (The three ODPs could correspond to the LSP expressed in this sentence. A modelling decision has to be taken according to the userÕs needs).
	2	*Water areas have names in natural language.*

Table A.6: LSP corresponding to *object property, datatype property* and *simple part-whole relation* ODPs

Symbols & Abbreviations	Description
AP<...>	Adjectival Phrase. It is defined as a phrase whose head is an adjective accompanied optionally by adverbs or other complements as prepositional phrases. AP is followed by the semantic role played by the concept it represents in the conceptual relation in question in <É>, such as e.g., *property.*
CATV	Verbs of Classification. Set of verbs of classification plus the preposition that normally follows them. Some of the most representative verbs in this group are: *classify in/into, categorize in/into, sub-classify in/into, group into, divide into.*
CD	Cardinal Number.
CN	Class Name. Generic names for semantic roles usually accompanied by preposition, such as *class, group, type, member, subclass, category, part, set, example* etc.
COMP	Verbs of Composition. Set of verbs meaning that something is made up of different parts. Some of the most representative ones are: *consist of, compose of, make up of, form of/by, constitute of/by.*
NP<...>	Noun Phrase. It is defined as a phrase whose head is a noun or a pronoun, optionally accompanied by a set of modifiers, and that functions as the subject or object of a verb. NP is followed by the semantic role played by the concept it represents in the conceptual relation in question in <...>, e.g., *class, subclass.*
PARA	Paralinguistic symbols like colon, or more complex structures as *as follows*, etc.
PREP	Prepositions
QUAN	Quantifiers such as *all, some, most, many, several, every,* etc.
REPRO	Relative pronouns such as that, which, whose.
()	Parentheses group two or more elements.
*	Asterisk indicates repetition.
[ ]	Elements in brackets are meant to be optional, which means that they can be present either at that stage of the sentence or not, and by default of appearance, the pattern remains unmodified.
¬	Elements preceded by this symbol should not appear in the pattern.

Table A.7: Symbols and abbreviations used in the formalization of LSPs

Bibliography

[AGM85] E. Alchourron, P. Gardenfors, and D. Makinson. On the logic of theory change: partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50(2), 1985.

[Ald05] Aldo Gangemi and Carola Catenacci and Massimiliano Ciaramita and Jos Lehmann. Ontology evaluation: A review of methods and an integrated model for the quality diagnostic task. Technical report, CNR, 2005.

[ATS+07] Richard Arndt, Raphael Troncy, Steffen Staab, Lynda Hardman, and Miroslav Vacura. Comm: Designing a well-founded multimedia ontology for the web. In *Proceedings of the 4th European Semantic Web Conference (ISCW'07)*, Busan Korea, November 2007. Springer.

[BGM05] Janez Brank, Marko Grobelnik, and Dunja Mladenić. A survey of ontology evaluation techniques. In *Proc. of 8th Int. multi-conf. Information Society*, pages 166–169, 2005.

[BH06] A. Budanitsky and G. Hirst. Evaluating WordNet-based measures of semantic distance. *Computational Linguistics*, 32(1):13 – 47, 2006.

[Blo05] Eva Blomqvist. Fully automatic construction of enterprise ontologies using design patterns: Initial method and first experiences. In Robert Meersman, Zahir Tari, Mohand-Said Hacid, John Mylopoulos, Barbara Pernici, Ězalp Babaoglu, Hans-Arno Jacobsen, Joseph P. Loyall, Michael Kifer, and Stefano Spaccapietra, editors, *OTM Conferences (2)*, volume 3761 of *Lecture Notes in Computer Science*, pages 1314–1329. Springer, 2005.

[BSZ03] P. Bouquet, L. Serafini, and S. Zanobini. Semantic coordination: A new approachand an application. In *Proc. of ISWC*, pages 130–145. LNCS, Springer Verlag, 2003.

[CLS05] P. Cimiano, G. Ladwig, and S. Staab. Gimme' the context: context-driven automatic semantic annotation with C-PANKOW. In *Proc. of WWW*, 2005.

[CV07] R.L. Calibrasi and P.M. Vitanyi. The Google Similarity Distance. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):370 – 383, 2007.

[dCGPMPSF08] G. Aguado 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 *Proc. of EKAW*, pages 32–47. LNCS, Springer Verlag, 2008.

[DdB+08] Ch. Le Duc, M. d'Aquin, J. Barrasa, J. David, J. Euzenat, R. Palma, R. Plaza, M. Sabou, and B. Villazon-Terrazas. Matching ontologies for context: The neon alignment plug-in. Deliverable D3.3.2, NeOn project, 2008.

[DEB+08] Klaas Dellschaft, Hendrik Engelbrecht, JosŐ Monte Barreto, Sascha Rutenbeck, and Steffen Staab. Cicero: Tracking design rationale in collaborative ontology engineering. In Sean Bechhofer, Manfred Hauswirth, JŽrg Hoffmann, and Manolis Koubarakis, editors, *ESWC*, volume 5021 of *Lecture Notes in Computer Science*, pages 782–786. Springer, 2008.

[dSM08] M. d'Aquin, M. Sabou, and E. Motta. Reusing Knowledge from the Semantic Web with the Watson Plugin. In *Demo session of International Semantic Web Conference, ISWC*, 2008.

[DV06] T. Declerck and M. Vela. Generic NLP Tools for Supporting Ontology Building. In *Proc. of LREC*, 2006.

[EdSZ07] J. Euzenat, M. d'Aquin, M. Sabou, and A. Zimmermann. Methods and tools for re-engineering non-ontological resources. Deliverable D3.3.1, NeOn project, 2007.

[Gan05] Aldo Gangemi. Ontology Design Patterns for Semantic Web Content. In *M. Musen et al. (eds.): Proceedings of the Fourth International Semantic Web Conference*, Galway, Ireland, 2005. Springer.

[GCB04] Aldo Gangemi, Carola Catenacci, and Massimo Battaglia. Inflammation ontology design pattern: an exercise in building a core biomedical ontology with descriptions and situations. In Domenico Maria Pisanelli, editor, *Ontologies in Medicine*. IOS Press, Amsterdam, 2004.

[GCCL06] Aldo Gangemi, Carola Catenacci, Massimiliano Ciaramita, and Jos Lehmann. Modelling ontology evaluation and validation. In *ESWC*, pages 140–154, 2006.

[GF94] M. Gruninger and M. Fox. The role of competency questions in enterprise engineering, 1994.

[GLW06] S. Ghilardi, C. Lutz, and F. Wolter. Did I damage my ontology? a case for conservative extensions in description logics. In Patrick Doherty, John Mylopoulos, and Christopher Welty, editors, *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR'06)*, pages 187–197. AAAI Press, 2006.

[GM08] J. Gracia and E. Mena. Web-Based Measure of Semantic Relatedness. In *Proc. of WISE*, pages 136 – 150. LNCS, Springer Verlag, 2008.

[GP07] Aldo Gangemi and Valentina Presutti. Ontology design for interaction in a reasonable enterprise. In Peter Rittgen, editor, *Handbook of Ontologies for Business Interaction*. IGI Global, Hershey, PA, November 2007.

[Gui05] G. Guizzardi. *Ontological foundations for structural conceptual models*. PhD thesis, University of Twente, Enschede, The Netherlands, Enschede, October 2005.

[GW04] Giancarlo Guizzardi and Gerd Wagner. A unified foundational ontology and some applications of it in business modeling. In *CAiSE Workshops (3)*, pages 129–143, 2004.

[Hay96] David C. Hay. *Data Model Patterns*. Dorset House Publishing, 1996.

[Hea92] M.A. Hearst. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proc. of the 14th International Conference on Computational Linguistics*, pages 539–545, 1992.

[Jua07] Juan Gomez-Romero and Fernando Bobillo and Miguel Delgado. An ontology design pattern for representing relevance in owl. In Karl Aberer, Key-Sun Choi, and Natasha Noy, editors, *The 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference 2007*, Busan, Korea, November 2007.

[Lew06] Holger Lewen. Topic-specific trust and open rating systems: an approach for ontology evaluation. In *Proceedings of WWW'06 4th International EON Workshop Evaluating Ontologies for the Web*, 2006.

[LMD$^+$09] V. Lopez, E. Motta, M. Dzbor, M. dÕAquin, S. Peroni, and D. Guidi. Final version of the question answering system. OpenKnowledge Deliverable D8.6, 2009.

[LTGP04] Adolfo Lozano-Tello and Asunción Gómez-Pérez. OntoMetric: A method to choose the appropriate ontology. *Journal of Database Management Special Issue on Ontological analysis, Evaluation, and Engineering of Business Systems Analysis Methods*, 15(2), 2004.

[MPdCGPSF08] E. Montiel-Ponsoda, G. Aguado de Cea, A. Gómez-Pérez, and M.C. Suárez-Figueroa. Helping Naive Users to Reuse Ontology Design Patterns. In *Proc. of the 1st International Workshop on Reuse and Reengineering over the Semantic Web*, 2008.

[MS02] Alexander Maedche and Steffen Staab. Measuring similarity between ontologies. In *Proc. of the 13th Conf. on Knowledge Engineering and Management*, 2002.

[NA05] Natasha Noy and Alan Rector. Defining N-ary Relations on the Semantic Web: Use With Individuals. Technical report, W3C, 2005. http://www.w3.org/TR/swbp-n-aryRelations/ (2004).

[Obe06] Daniel Oberle. *Semantic Management of Middleware*, volume I of *The Semantic Web and Beyond*. Springer, New York, FEB 2006.

[PBMW98] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford University, CA, USA, 1998.

[PG04] Semantic Web Best Practices and Deployment Working Group. Task force on ontology engineering patterns. description of work, archives, w3c notes and recommendations. http://www.w3.org/2001/sw/BestPractices/OEP/, 2004.

[PG08] V. Presutti and A. Gangemi. Content Ontology Design Patterns as Practical Building Blocks for Web Ontologies. In *Proceedings of the 27th International Conference on Conceptual Modeling (ER 2008)*, Berlin, 2008. Springer.

[PGD$^+$08] Valentina Presutti, Aldo Gangemi, Stefano David, Guadalupe Aguado de Cea, Mari-Carmen Suárez-Figueroa, Elena Montiel-Ponsoda, and María Poveda. Library of design patterns for collaborative development of networked ontologies. Deliverable D2.5.1, NeOn project, 2008.

[PM04] Robert Porzel and Rainer Malaka. A task-based approach for ontology evaluation. In *Proceedings of the ECAI-2004 Workshop on Ontology Learning and Population*, Sevilla, Spain, 2004.

[PST04] H. Pinto, S. Staab, and C. Tempich. Diligent: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engineering of ontologies. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)*, Valencia, Spain, 2004.

[RR04] Alan Rector and Jeremy Rogers. Patterns, properties and minimizing commitment: Reconstruction of the galen upper ontology in owl. In Aldo Gangemi and Stefano Borgo, editors, *Proceedings of the EKAW*04 Workshop on Core Ontologies in Ontology Engineering*. CEUR, 2004.

[SAd$^+$07] Marta Sabou, Sofia Angeletou, Mathieu d'Aquin, Jesus Barrasa, Klaas Dellschaft, Aldo Gangemi, Jos Lehmann, Holger Lewen, Diana Maynard, Dunja Mladenic, Malvina Nissim, Wim Peters, Valentina Presutti, and Boris Villazon. D2.2.1 methods for selection and integration of reusable components from formal or informal user specifications. NeOn Project Deliverable D2.2.1, The Open University, MAY 2007.

[SdM08a] M. Sabou, M. d'Aquin, and E. Motta. Exploring the Semantic Web as Background Knowledge for Ontology Matching. *Journal on Data Semantics*, XI, 2008.

[SdM08b] M. Sabou, M. d'Aquin, and E. Motta. Exploring the Semantic Web as Background Knowledge for Ontology Matching. *Journal of Data Semantics*, 2008.

[SG08] M. Sabou and J. Gracia. Spider: Bringing Non-Equivalence Mappings to OAEI. In *Proc. of the Third International Workshop on Ontology Matching*, 2008.

[SSG07] J. Sindhu, C. Sierra, and F. Giunchiglia. Trust and Reputation. OpenKnowledge Deliverable D4.7, 2007.

[Sva04] Vojtech Svatek. Design patterns for semantic web ontologies: Motivation and discussion. In *Proceedings of the 7th Conference on Business Information Systems*, Poznan, 2004.

[VD09] M. Vela and T. Declerck. Concept and Relation Extraction in the Finance Domain. In *Proc. of the 8th International Conference on Computational Semantics (IWCS-8)*, 2009.

[Vil09] B. Villazon. Methods and tools for re-engineering non-ontological resources. Deliverable D2.2.2, NeOn project, 2009.

[Vra05] Denny Vrandecic. Explicit knowledge engineering patterns with macros. In Chris Welty and Aldo Gangemi, editors, *Proceedings of the Ontology Patterns for the Semantic Web Workshop at the ISWC 2005*, Galway, Ireland, NOV 2005.

[Wat] Watson: Semantic web gateway. http://watson.kmi.open.ac.uk/.

[ZSdM08] F. Zablith, M. Sabou, M. d'Aquin, and E. Motta. Using Background Knowledge for Ontology Evolution. In *Proc. of the International Workshop on Ontology Dynamics (IWOD)*, 2008.