



NeOn: Lifecycle Support for Networked Ontologies

Integrated Project (IST-2005-027595)

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

D2.2.4 Final version of methods for re-engineering and evaluation

Deliverable Co-ordinator: Sofia Angeletou (OU)

Deliverable Co-ordinating Institution: Open Univesity (OU)

Other Authors: Holger Lewen (UKARL);Boris Villazón-Terrazas (UPM)

This deliverable describes the finalisation and evaluation of the methods and tools for evaluation and selection of knowledge components as well as non ontological resource re engineering within the context of the NeOn project.

Document Identifier:	NEON/2010/D2.2.4/v1.0	Date due:	January 31, 2010
Class Deliverable:	NEON EU-IST-2005-027595	Submission date:	January 31, 2010
Project start date	March 1, 2006	Version:	v1.0
Project duration:	4 years	State:	Final
		Distribution:	Public

NeOn Consortium

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

<p>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</p>	<p>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</p>
<p>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</p>	<p>Software AG (SAG) Umlandstrasse 12 64297 Darmstadt Germany Contact person: Walter Waterfeld E-mail address: walter.waterfeld@softwareag.com</p>
<p>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</p>	<p>Institut ‘Jožef Stefan’ (JSI) Jamova 39 SL-1000 Ljubljana Slovenia Contact person: Marko Grobelnik E-mail address: marko.grobelnik@ijs.si</p>
<p>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</p>	<p>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</p>
<p>Universität Koblenz-Landau (UKO-LD) Universitätsstrasse 1 56070 Koblenz Germany Contact person: Steffen Staab E-mail address: staab@uni-koblenz.de</p>	<p>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</p>
<p>Ontoprise GmbH. (ONTO) Amalienbadstr. 36 (Raumfabrik 29) 76227 Karlsruhe Germany Contact person: Jürgen Angele E-mail address: angele@ontoprise.de</p>	<p>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</p>
<p>Atos Origin S.A. (ATOS) Calle de Albarracín, 25 28037 Madrid Spain Contact person: Tomás Pariente Lobo E-mail address: tomas.pariantelobo@atosorigin.com</p>	<p>Laboratorios KIN, S.A. (KIN) C/Ciudad de Granada, 123 08018 Barcelona Spain Contact person: Antonio López E-mail address: alopez@kin.es</p>

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	11-09-2009	Holger Lewen	Setup the document and included my contribution on the TS-ORS evaluations.
0.2	17-11-2009	Boris Villazón-Terrazas	Included the Re-engineering methods chapter.
0.3	06-12-2009	Boris Villazón-Terrazas	Included the Evaluation of the Method and Patterns.
0.4	08-12-2009	Sofia Angeletou	First Complete Draft
0.5	09-12-2009	Sofia Angeletou	First Internal Revision
0.6	01-27-2010	Sofia Angeletou	Internal Revision Comments Amended and sent to internal QA for review

Executive Summary

In the present deliverable we describe the concluding of works carried out in the context of Workpackage 2, *Collaborative Aspects for Networked Ontologies*, in the duration of the NeOn project. The objectives of Workpackage 2 are to investigate and analyse the collaborative aspects in the development of networked ontologies. In addition to that, WP2 aims at describing methods and producing tools to support the collaborative ontology engineering activities.

This deliverable describes the finalisation of the works implemented in the scope of T2.2 *Methods and tools for collaborative engineering of ontologies*. In the past three years of the NeOn project T2.2 provided methods and tools to support re-engineering, evaluation and selection of ontologies and their components. These works have been reported in the previous related deliverables [SAd⁺07b], [VTAGS⁺08] and [SdCd⁺09].

In this deliverable we present the evaluation and finalisation of the methods described in the past three years. In particular:

- Chapters 2 and 3 present the work of UKARL on the performance improvement of their Topic Specific Open Rating System (Chapter 2), and a simulation to determine the behavior of the algorithm in certain test cases (Chapter 3).
- Chapters 4 and 5 present the updated work of UPM on the method for re-engineering non-ontological resources, and the patterns for re-engineering non-ontological resources. In addition they present a user based evaluation on the effectiveness of these methods and set of patterns.
- Finally, Chapter 6 presents the work of the OU on the refinement on folksonomy re-engineering and the evaluation of the knowledge sources used for this purpose.

Contents

1	Introduction	11
1.1	Deliverable Structure	12
1.2	Integration with other Workpackages of NeOn	12
2	Improvements and Performance Evaluation of the Topic-Specific Open Rating System	13
2.1	Introduction	13
2.2	Performance Limiting Factors in the Old Code and Reengineering	13
2.2.1	Changes in Architecture	14
2.2.2	UML-Diagram	14
2.2.3	Interaction with other Programs	15
2.3	Complexity Analysis	18
2.3.1	Complexity of Trust Computation	18
2.3.2	Runtime Complexity	19
2.3.3	Further Optimization	20
2.4	Evaluation	20
2.4.1	Results	20
2.5	Result Analysis	32
2.5.1	Meta-trust Propagation and Trust Computation	32
2.5.2	Overall Computation	32
2.5.3	Review Retrieval	35
2.6	Results Learnt from Benchmark	37
3	Simulation-based Evaluation of the Topic-Specific Open Rating System	38
3.1	Introduction	38
3.2	Setup	38
3.2.1	System	38
3.2.2	Basic Experiment Setup	39
3.2.3	Variations During Test-Runs and Alteration of the Setup	40
3.2.4	Remarks on Implementation	41
3.2.5	Result Generation	41
3.3	Results	42
3.3.1	Expected Results	42
3.3.2	Scenario 1	42
3.3.3	Scenario 2	43
3.3.4	Scenario 3	48
3.3.5	Comparison 5% Coverage with 100 and 1000 Users, 0% Error	48

3.3.6	Minimal Setting with Expected Behavior	54
3.4	Conclusion	58
3.4.1	Validity of Simulation Results for Real World Systems	58
3.4.2	Accuracy of Ranking Results for the Individual	58
3.4.3	Significance of Error Settings	58
3.4.4	Attacks on the System	59
3.4.5	Lessons Learned	59
4	Method for Re-engineering Non-Ontological Resources into Ontologies	60
4.1	Model for Re-engineering Non-Ontological Resources	61
4.2	Non-ontological Resources Re-engineering Process	62
4.3	Patterns for Re-engineering NORs into Ontologies	63
4.3.1	Transformation approaches	64
4.3.2	Formalization of the resources	65
4.3.3	Semantics of the relations among the non-ontological resource entities	67
4.3.4	Template for the PR-NOR	67
4.4	PR-NOR Library	68
4.4.1	Patterns for Re-engineering Classification Schemes into Ontologies	68
4.4.2	Patterns for Re-engineering Thesauri into Ontologies	90
5	Evaluation of the Method and Patterns for Re-engineering Non-Ontological Resources	102
5.1	Supporting Non-ontological Resource Re-engineering	102
5.1.1	Overview and Objectives	102
5.1.2	Assumptions and user study setup	102
5.1.3	Finding and observations	103
5.1.4	Further analysis and discussion	103
5.2	Testing the PR-NOR Software Library	104
5.2.1	Overview and Objectives	104
5.2.2	Assumptions and user study setup	105
5.3	Findings and observations	107
5.3.1	User study 1: Time/effort spent, and quality of the resultant ontology, by using a classification scheme	107
5.3.2	User study 2: Time/effort spent, and quality of the resultant ontology, by using a thesaurus	107
5.3.3	User study 3: Quality of the resultant ontology	108
5.3.4	User study 4: User Satisfaction	109
5.4	Further analysis and discussion	110
5.4.1	Analysis of the user study 1	110
5.4.2	Analysis of the user study 2	110
5.4.3	Analysis of the user study 3	110
5.4.4	Analysis of the user study 4	111
5.4.5	Identified strengths and weaknesses	111
5.4.6	Prospective further work	112
6	Improvements of the Folksonomy Enrichment Algorithm	113
6.1	Introduction	113
6.1.1	Background and Motivation	113

6.1.2	Overview	115
6.2	Evaluation of Knowledge Sources	115
6.2.1	Building and Querying Knowledge Structures	115
6.2.2	Experiments	118
6.2.3	Conclusions	124
6.3	Second Version of the FLOR Folksonomy Enrichment Algorithm	124
6.3.1	FLOR Algorithm	124
6.3.2	FLOR Ontology	126
6.3.3	Lexical Processing of Tags	127
6.3.4	Sense Selection for Tags	128
6.3.5	Semantic Aggregation of Senses	132
6.4	Ongoing Work	135
	Bibliography	136

List of Tables

4.1	Pattern for Re-engineering Non-Ontological Resource Template	67
4.2	Pattern for re-engineering a classification scheme, which follows the path enumeration data model, into an ontology schema.	69
4.3	Pattern for re-engineering a classification scheme, which follows the adjacency list data model, into an ontology schema	71
4.4	Pattern for re-engineering a classification scheme, which follows the snowflake data model, into an ontology schema.	74
4.5	Pattern for re-engineering a classification scheme, which follows the flattened data model, into an ontology schema.	77
4.6	Pattern for re-engineering a classification scheme, which follows the path enumeration data model, into an ontology.	80
4.7	Pattern for re-engineering a classification scheme, which follows the adjacency list data model, into an ontology.	82
4.8	Pattern for re-engineering a classification scheme, which follows the snowflake data model, into an ontology.	85
4.9	Pattern for re-engineering a classification scheme, which follows the flattened data model, into an ontology.	87
4.10	Pattern for re-engineering a term-based thesaurus, which follows the record-based data model, into an ontology schema	90
4.11	Pattern for re-engineering a term-based thesaurus, which follows the relation-based model, into an ontology schema	93
4.12	Pattern for re-engineering a term-based thesaurus, which follows the record-based data model, into an ontology	96
4.13	Pattern for re-engineering a term-based thesaurus, which follows the relation-based model, into an ontology	99
6.1	Quantitative results of the enrichment evaluation	121
6.2	User Questions and Responses	121
6.3	Different Types of Tag Isolators	128

List of Figures

2.1	This graphic depicts the new database schema of the TS-ORS.	14
2.2	This graphic depicts the updated UML class diagram of some of the TS-ORS core components. 16	
2.3	This graphic depicts the updated UML class diagram of the rest of the TS-ORS core components. 17	
2.4	This graphic depicts the new UML class diagram of the servlets used for interaction with Cupboard.	18
2.5	This graphic depicts the results of the benchmark of the computation and meta trust propagation. Data is presented as time in seconds. (Run on MacBook Pro)	22
2.6	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. 500 runs were performed and only the top review was considered for the computation. (Run on MacBook Pro)	23
2.7	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. 500 runs were performed and only the top 3 reviews was considered for the computation. (Run on MacBook Pro)	24
2.8	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. 500 runs were performed and all reviews were considered for the computation. (Run on MacBook Pro)	25
2.9	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. 50 runs were performed. (Run on MacBook Pro)	26
2.10	This graphic depicts the results of the benchmark of the computation and meta trust propagation. Data is presented as time in seconds. (Run on Intel Core 2 Quad)	27
2.11	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. 500 runs were performed and only the top review was considered for the computation. (Run on Intel Core 2 Quad)	28
2.12	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. 500 runs were performed and only the top 3 reviews was considered for the computation. (Run on Intel Core 2 Quad)	29
2.13	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. 500 runs were performed and all reviews were considered for the computation. (Run on Intel Core 2 Quad)	30
2.14	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. 50 runs were performed. (Run on Intel Core 2 Quad)	31
2.15	This graphic presents a direct comparison between the two versions of the code and the two different computers. Please note that the left axis has a logarithmical scale.	33
2.16	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)	34

2.17	This graphic presents the distribution of time spend for the different tasks during the trust computation for a different number of threads.	34
2.18	This graphic presents a direct comparison between the two versions of the code and the two different computers. Please note that only min times are compared and the timer is in ms.	36
3.1	First Scenario with 0% Error	44
3.2	First Scenario with 10% Error	45
3.3	First Scenario with 20% Error	46
3.4	Second Scenario with 0% Error	47
3.5	Second Scenario with 10% Error	49
3.6	Second Scenario with 20% Error	50
3.7	Third Scenario with 0% Error	51
3.8	Third Scenario with 10% Error	52
3.9	Third Scenario with 20% Error	53
3.10	Comparison of Results Scenario 10% Error	55
3.11	For this graph, we have checked for each ontology in the investigated group, which percentage of the users in the group investigated has no local trust information for this ontology. The findings were sorted into 10% buckets. The computations were performed with 100 users, 5% coverage and 0% error.	56
3.12	For this graph, we have checked for each ontology in the investigated group, which percentage of the users in the group investigated has no local trust information for this ontology. The findings were sorted into 10% buckets. The computations were performed with 1000 users, 5% coverage and 0% error.	57
4.1	Re-engineering Model for Non-Ontological Resources	61
4.2	Re-engineering process for Non-Ontological Resources	62
4.3	Transformation approaches	64
5.1	Personal assessment about the ontology developed with less effort/time	107
5.2	Personal assessment about the quality of the ontologies	108
5.3	Personal assessment about the ontology developed with less effort/time	108
5.4	Personal assessment about the quality of the ontologies	109
5.5	Personal assessment about the quality of the ontologies	109
5.6	The results of SUMI questionnaires for the PR-NOR Software Library	110
6.1	Part (a) represents a folksonomy tagspace (tags are textual labels). Part (b) represents the semantic layer over this tagspace as a result of the folksonomy enrichment	114
6.2	Result screenshot for the query <i>sport</i> in system S1.	120
6.3	Increase in results on the user entered keywords	123
6.4	Example of FLOR enrichment for the tagset of Resource_X	125
6.5	The updated version of FLOR Enrichment Algorithm	125
6.6	Ontology of FLOR	127
6.7	An example Semantic Entity returned from Watson when querying for <i>Lake</i>	130
6.8	Two Senses Sharing Semantic Entities (SE) provides evidence on their relations	134

Chapter 1

Introduction

The objectives of Workpackage 2, *Collaborative Aspects for Networked Ontologies*, are to investigate and analyse the collaborative aspects in the development and reuse of networked ontologies. In addition to that, WP2 aims at describing methods and producing tools to support the collaborative ontology engineering activities. The task T2.2, in particular, describes methods and tools to support re-engineering, evaluation and selection of ontologies and ontological components in the process of networked ontology engineering. As a result, the work performed in task T2.2. has been divided in two subtasks, one focusing on the reuse and re-engineering and the other focusing on evaluation and selection.

T2.2.a : Methods and tools for ontology evaluation and selection This task focuses on the development of methods and tools that support the evaluation and selection of ontological components in the context of building networked ontologies. The works carried out in this task during the past three years include the collaborative evaluation of ontology modules with the use of Open Rating Systems. This work has been reported in [SdCd⁺09].

T2.2.b : Methods and tools for re-engineering non-ontological resources. This task focuses on the reuse and re-engineering of knowledge resources. During the last three years of the project the works carried out include the development of re-engineering patterns for transforming non-ontological resources to ontologies and the implementation of methods for re-engineering folksonomies to ontologies. These works have been reported in [VTAGS⁺08]

The present deliverable describes the concluding efforts on the works of different partners in T2.2 both in T2.2.a and T2.2.b. The works presented here are underlined by the evaluation and refinement of their methods utilising different types of evaluations.

At first the work carried out by UKARL aims at improving the performance of the Topic-Specific Open Rating System (TS-ORS) taking into account the results of the last deliverable [SdCd⁺09]. The improvement process, the modified architecture and additional features are described. The evaluation strategy used by the UKARL was to use simulation for the Topic-Specific Open Rating System (TS-ORS) in addition to performance tests. The main purpose of the simulation was to check the system behavior in a controlled environment and draw conclusions for its use in real world systems.

Second, the UPM improved the method and the patterns for re-engineering non-ontological resources presented in [VTAGS⁺08]. In addition, they carried out a user based evaluation on using the method and the set of patterns developed for re-engineering. The goal of this evaluation was to gain evidence on whether this method and the usage of patterns can effectively support and improve the work of users with non-extensive ontology development knowledge.

Finally, the OU worked towards the improvement of the folksonomy enrichment algorithm (FLOR) presented in [VTAGS⁺08]. The main outcome of the evaluation in the last deliverable was the low enrichment rate of the previous version of FLOR due to the restrictive disambiguation techniques. This deliverable describes the improvements of the FLOR algorithm as well as the addition of a method that returns the integrated semantic

descriptions for a given tag space. In addition, the comparison of the two main knowledge sources that affect the enrichment process is described.

1.1 Deliverable Structure

The contributions of this deliverable are presented in the following:

Chapter 2 describes which steps were taken, how the architecture changed, and also how the performance improved after rewriting large parts of the code. Also, additional features are described.

Chapter 3 describes a simulation for the Topic-Specific Open Rating System (TS-ORS) in addition to performance tests. To achieve this UKARL has run the same task several times, altering parameters in order to see how different coverage or errors made by the users affects the outcome.

Chapter 4 presents an updated version of the method for re-engineering non-ontological resources into ontologies. More precisely, the model, the process and the patterns for re-engineering non-ontological resources are updated and improved.

Chapter 5 provides qualitative and quantitative evidence that using the improved method for re-engineering non-ontological resources into ontologies, along with the set of patterns, leads to users being able to design ontologies faster and/or better quality standards.

Chapter 6 describes the comparison of the knowledge sources used during the FLOR enrichment and the improved version of the enrichment algorithm.

1.2 Integration with other Workpackages of NeOn

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

WP1 The result ontologies, after the re-engineering process, will follow the networked model proposed in WP1. In addition, the improved TS-ORS will be included in the CupBoard system.

WP5 The methods described in this deliverable are included in the NeOn Methodology for building ontology networks

WP6 The methods are (being) implemented to plugins and integrated into the NeOn ToolKit.

WP7, WP8 Methods and tools described in this deliverable applied to the NeOn Project use cases.

Chapter 2

Improvements and Performance Evaluation of the Topic-Specific Open Rating System

2.1 Introduction

Taking the source and performance measures published in the last deliverable [SdCd⁺09], we have rewritten large parts of the code to make the application run faster and with less memory consumption. In this report we will describe which steps were taken, how the architecture has improved, and also how the performance has improved with the new code base. Also, the new features are described. The new code base will also be used in our Cupboard system [dL09].

2.2 Performance Limiting Factors in the Old Code and Reengineering

After profiling the old code, the main limiting factors were identified as database access and the implementation of multithreading. As a first step, the database connection pool was replaced by a faster one, which yielded some speed improvement, but was not sufficiently satisfying. The main speedup occurred when we started bundling the statements writing to the database, i.e., caching the statements inserting data into the database in a StringBuffer and then writing a huge insert instead of a couple of thousand small insert statements. Because this way less connections have to be requested from the connection pool, performance increases notably.

In terms of memory consumption, most memory is consumed by the big matrices we use for trust computation. The former approach for parallelization had multiple instances of a thread perform the same computations on different data. Taken one computation as a baseline, each additional computation run in parallel took up the same memory as the initial thread. Running four threads thus meant quadrupling the memory needed. We have changed the computations to now parallelize the computations by distributing parts of the computation over more threads. Whereas formerly the matrix multiplication step would have run on one thread, but with more threads performing the same multiplication on different data in parallel, we now run one matrix multiplication but distribute it over more threads. Since all threads can work on the same matrix, there is no increase of memory consumption. Whenever possible, we reengineered the core methods of the algorithm to support multithreading.

For the meta-trust propagation, we redesigned the algorithm so that the propagation is done directly in the database, without many reads and writes from the java program. This also caused a significant speed-up. Keep in mind that the meta-trust propagation is in essence taking trust statements with a bigger scope and transforming them into smaller, more specific trust statements. To do this, it is first established whether or not a trust statement exists for the scope covered by the meta-trust statement then, where no trust statement exists, the value for the meta-trust statement is propagated. The detailed explanation can be found in [SdCd⁺09].

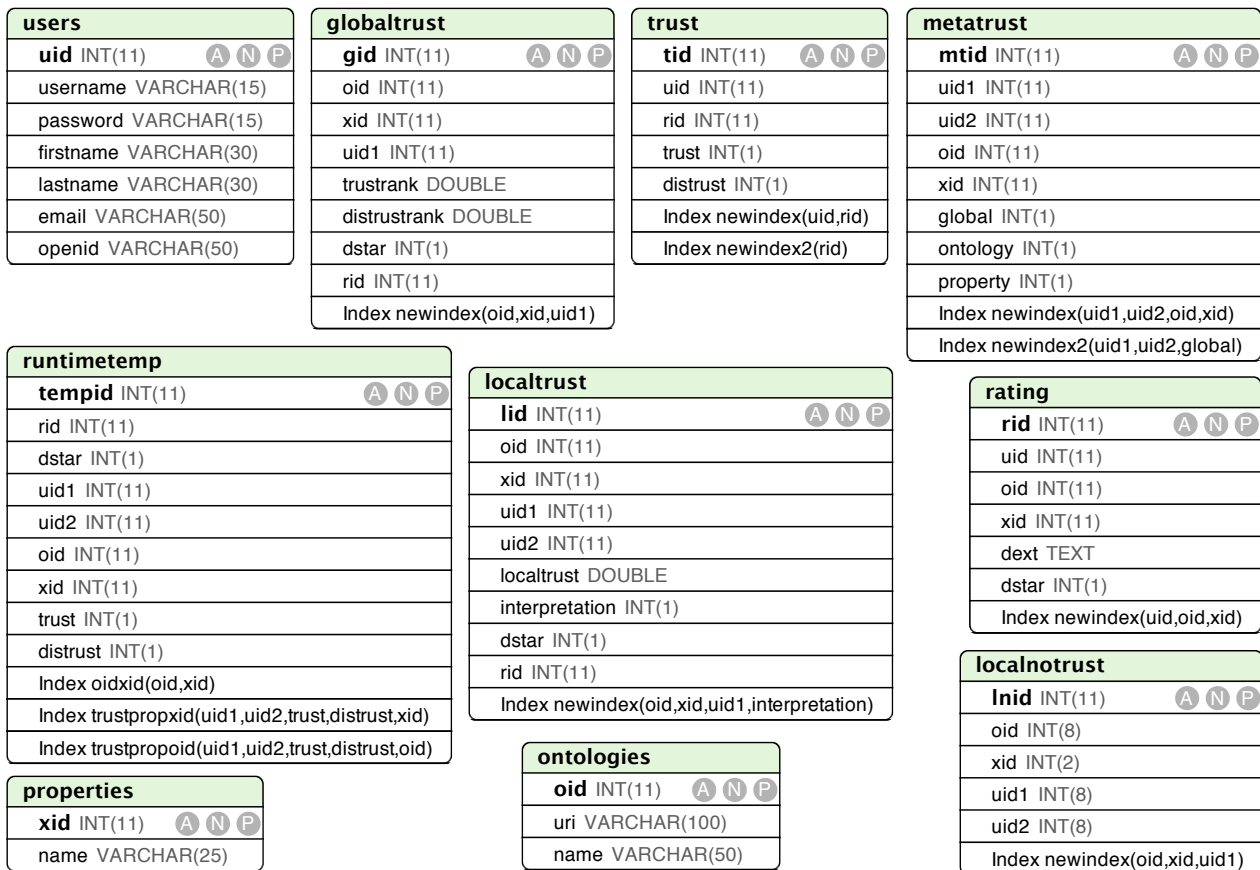


Figure 2.1: This graphic depicts the new database schema of the TS-ORS.

2.2.1 Changes in Architecture

We have updated both the database schema, and the structure of the source code.

Database Schema

While most of the schema has stayed the same, we have introduced a new table *localnotrust* where information is stored about which users are not connected locally. This table is then used for a faster information lookup at runtime. The updated schema can be found in Figure 2.1. We have also updated the index structure to improve performance. As before, when the recomputation is triggered during runtime, temp versions of the tables *runtime temp*, *localtrust*, *globaltrust*, and *localnotrust* are created. When the computations are completed, they replace the tables used at runtime.

2.2.2 UML-Diagram

Because of the performance optimization explained above, also the design of a number of classes and methods has changed. The updated UML Class-Diagrams can be found in Figure 2.2 and Figure 2.3. We will now lay out the changes and new functionality.

Changes to TS-ORS Core

One of the most important changes is the redesign of the multithreading functionality. The class *Multithreading* was removed, and the Jama *Matrix* class extended by methods that allow multithreading. The new *Caching* class allows to cache results based on standard parameters. The *computations* class was cleaned up by moving small methods into bigger ones, and extended with a method to retrieve the trust statistics for a review.

2.2.3 Interaction with other Programs

The number of Java Servlets used for the integration with Cupboard has been increased to offer access to the new functionality of retrieving the trust statistics for a review, and also to trigger the caching mechanism. The new UML class diagram can be seen in Figure 2.4.

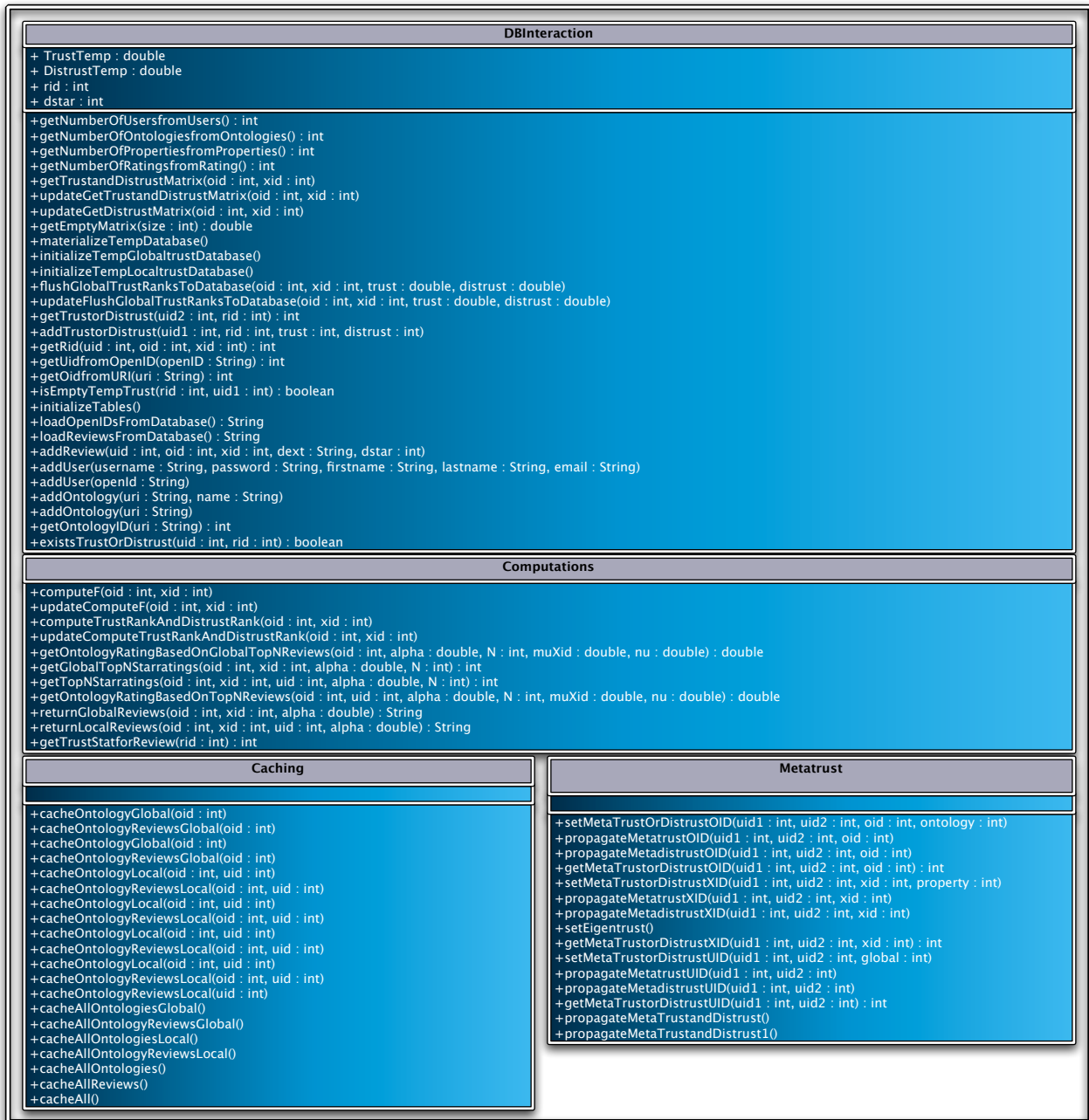


Figure 2.2: This graphic depicts the updated UML class diagram of some of the TS-ORS core components.

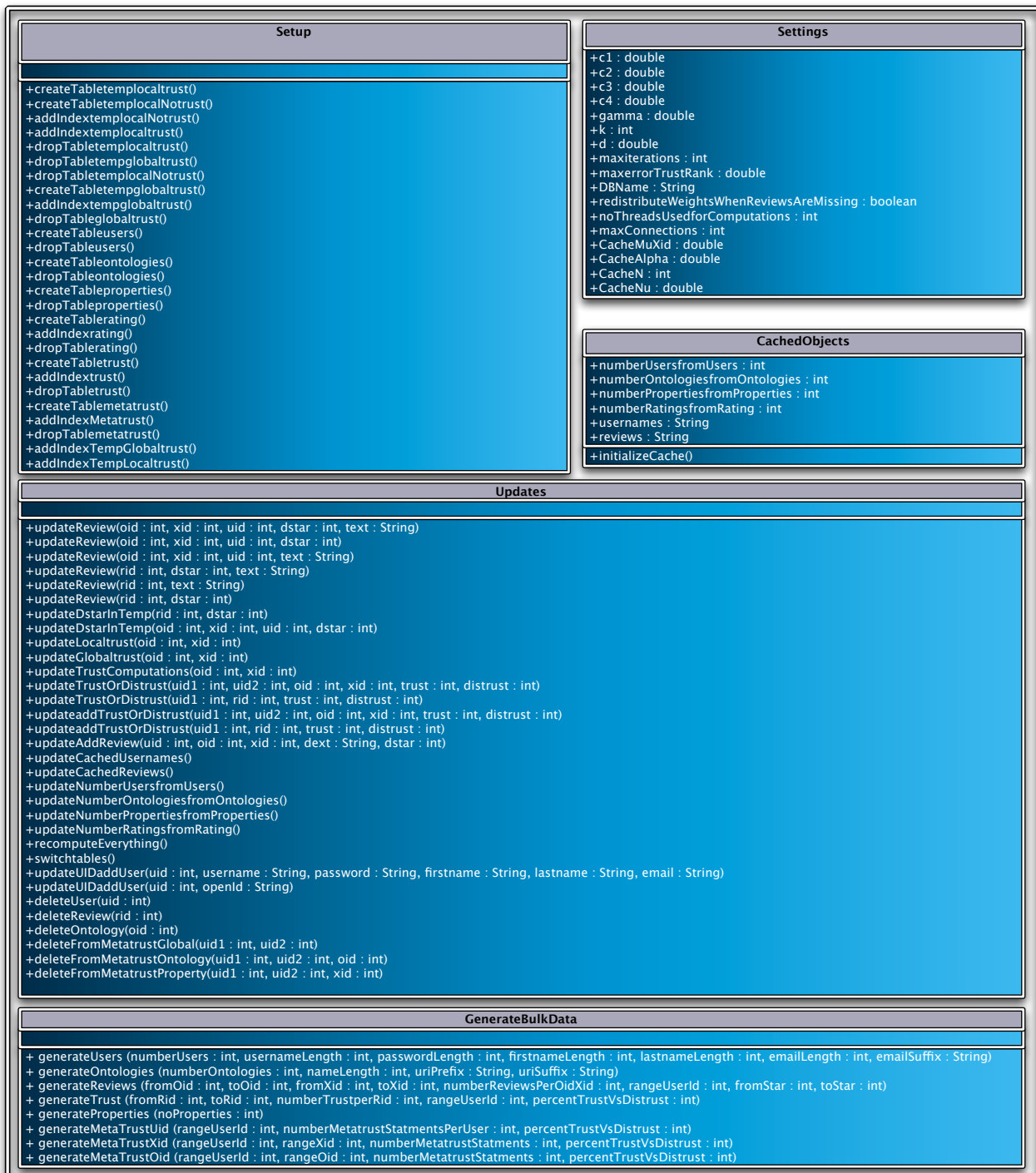


Figure 2.3: This graphic depicts the updated UML class diagram of the rest of the TS-ORS core components.

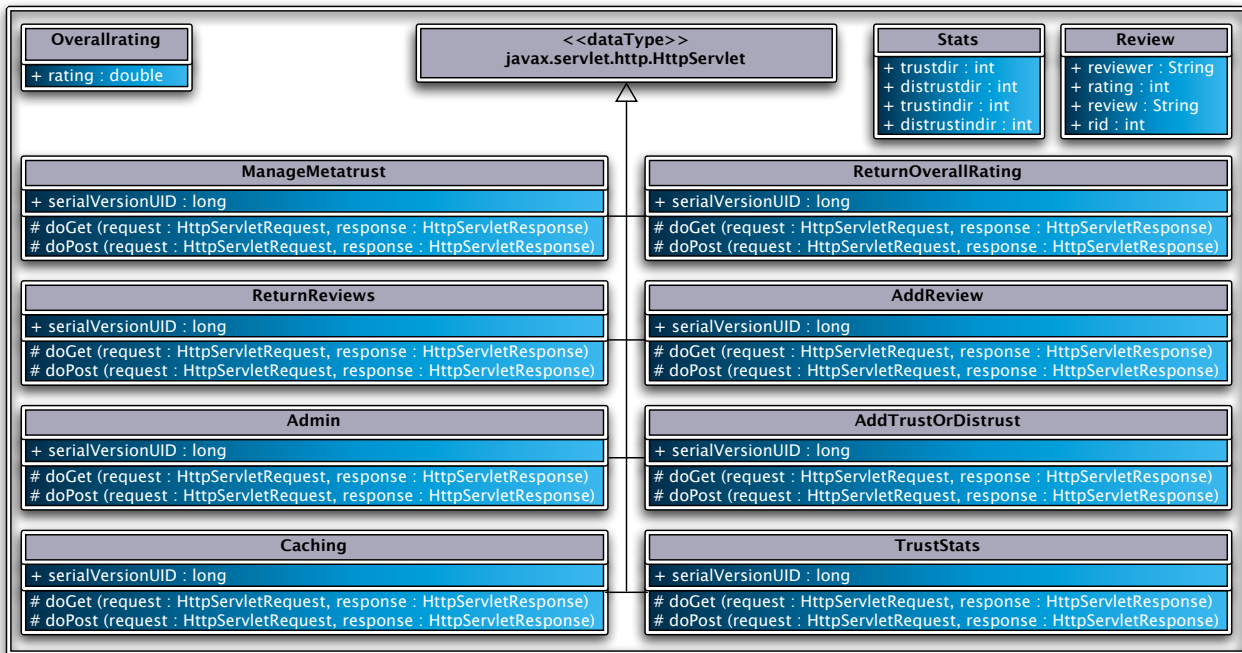


Figure 2.4: This graphic depicts the new UML class diagram of the servlets used for interaction with Cupboard.

2.3 Complexity Analysis

In order to understand the behavior examined during the benchmarks, it is also important to analyze the complexity of the algorithms. We have to distinguish between the computations run offline (the trust computation), and the computations needed at runtime (ranking of reviews, computing an overall score for an ontology). We will start with the computations performed offline. We use the common Big-O notation [Knu76].

2.3.1 Complexity of Trust Computation

Since the Big-O notation is dominated by the part of the algorithm having the highest complexity, we will analyze each step taken separately to determine the Big-O complexity. Since constants are irrelevant for the complexity analysis, we do not try to provide the exact number of times an operation is performed, but concentrate on the complexity of the operation performed. Usually one input parameter is chosen, since the result is easier to understand than trying to combine all variables.

In our system, we have a number of variables. The number of ontologies, the number of properties, the number of users, the number of reviews, and the number of trust statements on these reviews. We will consider the number of users as our input parameter, since it is the one having the largest impact on the complexity. It is important to understand that in the default setting the computations are performed exactly $\#ontologies * \#properties$ times (since the trust is computed for each ontology–property combination individually). This means that if we have 100 ontologies with 5 properties each, we have to compute the results for all $100 * 5 = 500$ combinations.

When the offline computations are triggered, the following operations are used:

- Matrix addition (in $O(n^2)$)
- Matrix subtraction (in $O(n^2)$)
- Matrix transposition (in $O(n^2)$)

- Matrix multiplication (in $O(n^3)$)¹
- Matrix scalar multiplication (in $O(n^2)$)
- The majority rounding (in $O(n^3)$) which consists of n times the following:
 - Sorting (in $O(n * \log n)$)
 - Interpreting Trust (in $O(n^2)$)
- TrustRank computation (in $O(n^2)$)
- DistrustRank computation (in $O(n^2)$)

Since the performed database operations all have a complexity below $O(n^3)$, the overall complexity of the computations is in $O(n^3)$. As long as there are not too many users, there is no problem computing everything for every ontology and property separately. If the time taken to compute the trust and distrust is taking too long for the needs of the application, the number of times the computation is performed can be reduced by grouping the available trust information and only perform the computations where they promise the most gain. Possible groupings could for example be by property or by ontologies of a special area.

2.3.2 Runtime Complexity

Here we have to distinguish two tasks the TS-ORS performs at runtime: Ranking the reviews for a given ontology–property combination, and computing an overall rating for an ontology. We also have to distinguish the cases where the user is identified (and local trust can be used), and the case where the user is unknown (and global trust has to be used).

Ranking of Reviews

For this task the system is given as input parameters the ontology and property for which the reviews have to be retrieved, and the user who is requesting the information (if available). Also the combination of trust and distrust can be influenced by a parameter. For the result based on the global reviews, the method basically just performs one database query and then provides the ordered reviews as output. Its runtime is dependent on the number of reviews that exist for this ontology–property combination. Within the database query, the results are ordered. So with n being the number of reviews, the complexity of the retrieval is in $O(n * \log n)$ which is mainly due to the sorting needed. In case the user is known, more database queries are performed, but the complexity stays in $O(n * \log n)$ with n being the number of existing reviews. Since there cannot be more than 1 review per user for each ontology–property combination, the number of users is an upper bound for the number of reviews. Most of the times, the number of reviews will be far smaller than the number of users.

Overall Rating of Ontologies

In case only the overall rating of an ontology has to be computed, the complexity is the same as for the ranking of the reviews, since for all properties of an ontology, the top n reviews have to be retrieved. Since during this process, again the results are sorted in the database, the complexity is $O(n * \log n)$ for both retrieval based on global and on local trust, with n being the number of reviews. Since the number of properties is a constant factor in any installation of the system, it does not influence the complexity. For runtime performance it is important to know that limiting the number of top reviews based on which the overall rating is computed does lower the time needed, but does not affect the theoretical complexity. Again, the number of users is an upper bound for n .

¹We are aware that other methods for matrix multiplication exist that have a lower complexity, but their overhead is too big to make them useful in our case

2.3.3 Further Optimization

While it is not possible to lower the worst case complexity, we have tested successfully an optimization for scenarios, where many users are not connected to the web of trust for a given ontology–property combination. Imagine the case where there are a million users in the system, but only a few reviews and trust statements on these reviews. Running the computation with a million times a million user matrix would result in a serious performance problem. Luckily, we managed to address this case by reducing the number of users taken into account for the computation to those actually affecting the outcome of the computation. By the nature of the algorithms, the only users actually affecting the outcome of the computation are the users who wrote the reviews for that ontology–property combination, and the users that have made a (dis)trust (or meta-(dis)trust) statement covering one of these reviews. In order to only use the data necessary, we first get the IDs of all users that have either written or (dis)trusted reviews, and fill two hash maps with a mapping from original ID to new ID and vice versa. The new IDs start with 1 and are auto incremented. The trust and distrust matrix are then filled with the data using the new IDs, and after the computation is done, data is written back using the old IDs. The result is exactly the same as if the algorithms were run on the complete user-base. The overhead is very small, since writing and reading from the Hashtable can be done in almost constant time. If we use this technique, then we can abandon the *localnotrust* table, since we would not have an entry for all the users that we did not consider for the computation. In our algorithm at runtime, we derive these users instead by first seeing which authors wrote reviews for that ontology–property combination, and who of them is not trusted by our user under consideration. This also saves space in the database.

Using the optimization, we can ensure that the complexity is in $O(m^3)$, with m being the number of users who wrote or (dis)trusted reviews. In the worst case, $m = n$, but in the normal data-sparse setting, we can save time during the computation.

2.4 Evaluation

In order to make the benchmarking results comparable to the old benchmark featured in deliverable [SdCd⁺09], we have left the evaluation methods unchanged and have just run it using the new code. So the following diagrams will be the same as in the older deliverable, just with updated times. The Setup is exactly as described in [SdCd⁺09]. In order to compute a more accurate average time, we have run the overall computation task 500 times instead of 300 times and also changed the time measurement to use Java nano time, which provides more accurate results. We did not use the optimization described in Section 2.3.3, because our scenarios were not data sparse.

2.4.1 Results

We have performed the benchmark again on two systems: A Dualcore MacBook Pro running Mac OS X 10.5.8 and a QuadCore PC this time running Ubuntu 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 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 or distrusting each of these reviews, one with a 50%–50% distribution and a worst case scenario (100%–100%). Worst case means that every 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%. We have furthermore assumed that of the users making trust-statements, 70% of the trust statements were trust, and 30% were distrust. For the meta-trust generation, we have fixed the percentage of global, ontology- and property-specific meta trust statements to 20% per user for all runs, i.e. each users meta trusts 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. As shown in section 2.3 the complexity of the computation does not increase by having more ontologies, since the ontologies are 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 10.5.8. The harddisk is a ST9320421ASG (320GB 7200 RPM, 16MB Cache, avg seek time 11 ms). This time we decided to use Java 6 64bit with Eclipse Version is 3.5 for Mac. MySQL versions are 5.1.36 64bit for the server and 5.1.8 for the J-Connector.

The results are shown in the Figures 2.5, 2.6, 2.7, 2.8 and 2.9. The percentages in the labels refer to the percentage of reviews, trust- and meta trust 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 meta trust 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 Ubuntu 64bit

The computation was performed on a 2.40 GHz Intel Core 2 Quad Q6600 processor with 8GB 800 MHz DDR2 SDRAM running Ubuntu Kernel 2.6.28-15 x86_64. The harddisk is a Samsung HD103UJ (1TB, 7200 RPM, 32MB Cache, avg seek time 8.9). Eclipse Version is 3.5.0 for Linux 64bit. Java version was OpenJDK Runtime Environment (IcedTea6 1.4.1) 6b14-1.4.1-0ubuntu11. We used MySQL 5.4.1 beta linux x86_64 ICC-GLIBC23 for the server and 5.1.8 for the J-Connector.

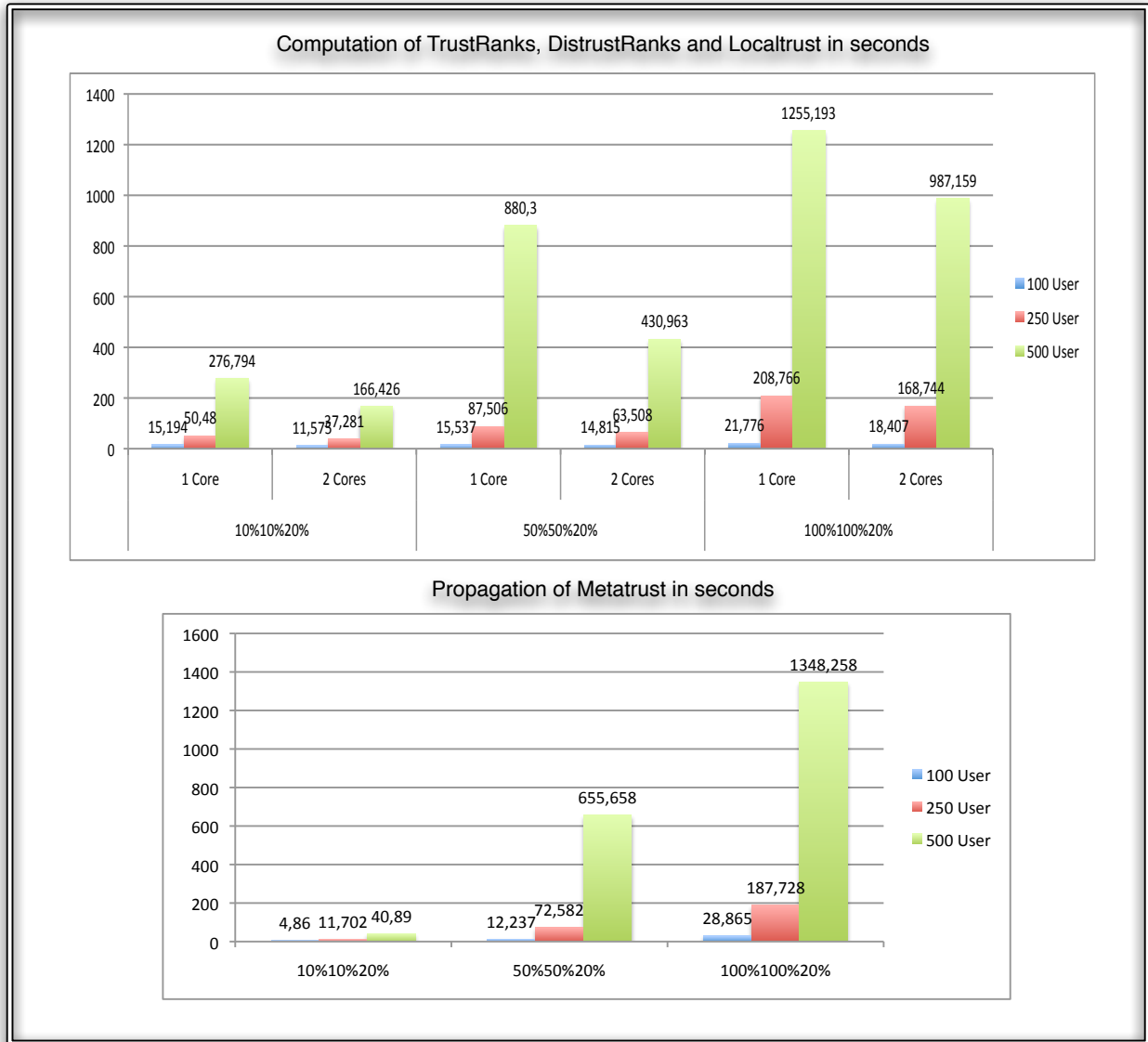


Figure 2.5: This graphic depicts the results of the benchmark of the computation and meta trust propagation. Data is presented as time in seconds. (Run on MacBook Pro)

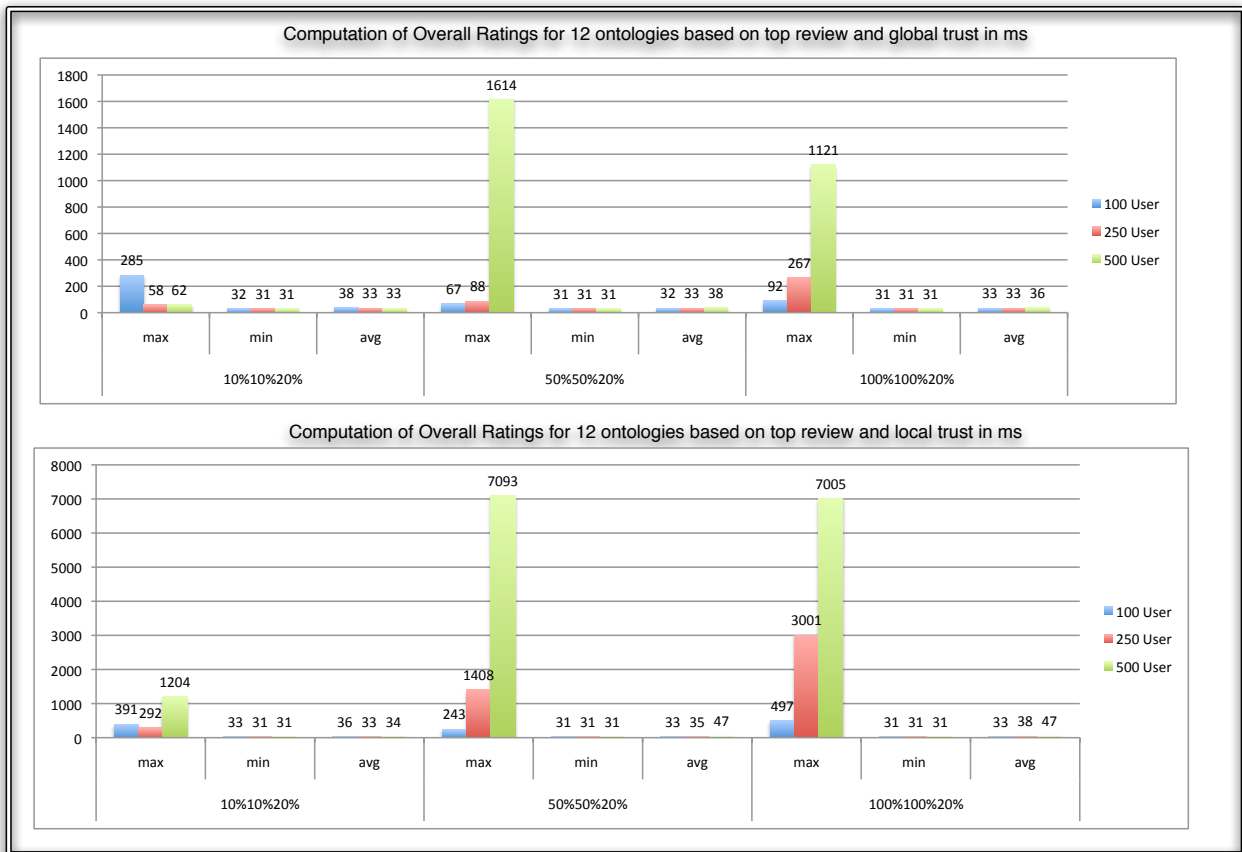


Figure 2.6: This graphic presents the times taken (min, max and avg) for returning the overall rating for all 12 ontologies in ms once for global and once for local trust. 500 runs were performed and only the top review was considered for the computation. (Run on MacBook Pro)

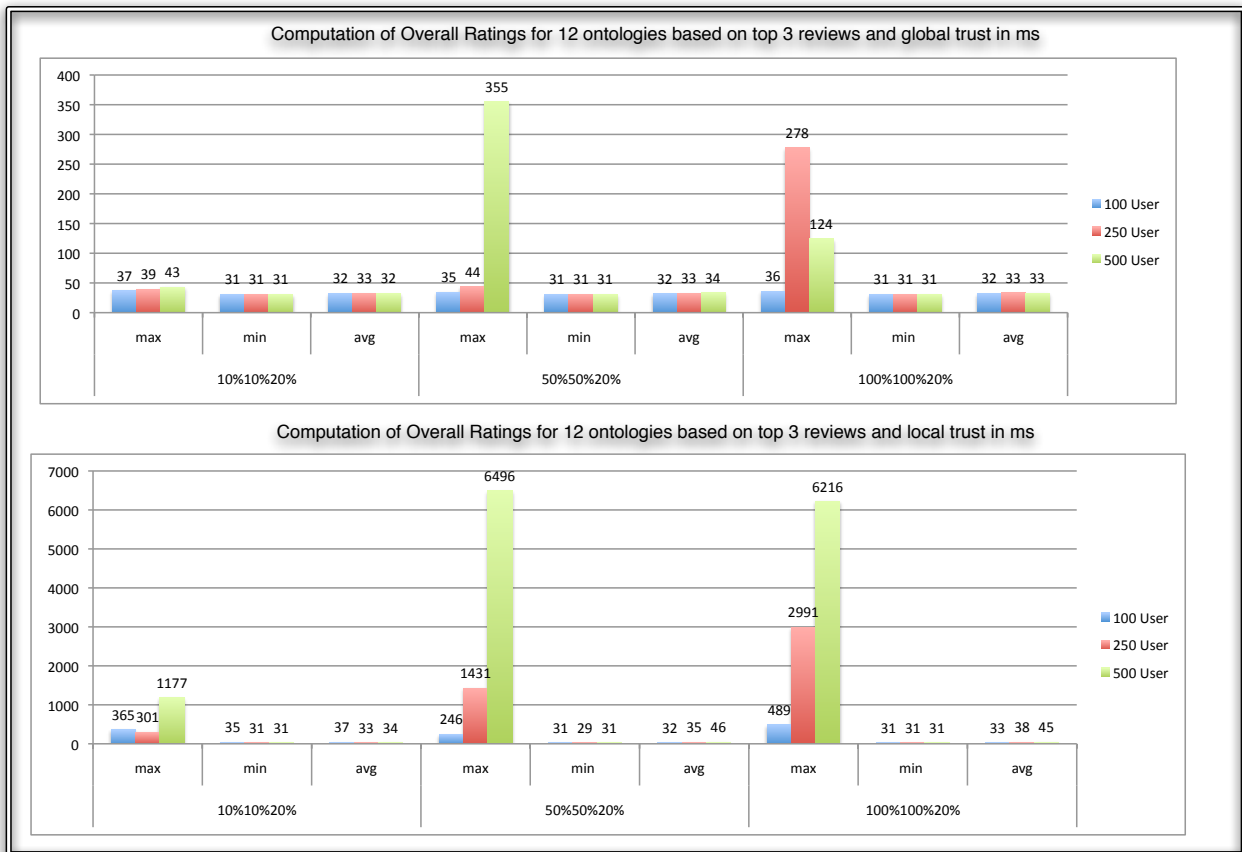


Figure 2.7: This graphic presents the times taken (min, max and avg) for returning the overall rating for all 12 ontologies in ms once for global and once for local trust. 500 runs were performed and only the top 3 reviews was considered for the computation. (Run on MacBook Pro)



Figure 2.8: 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. 500 runs were performed and all reviews were considered for the computation. (Run on MacBook Pro)

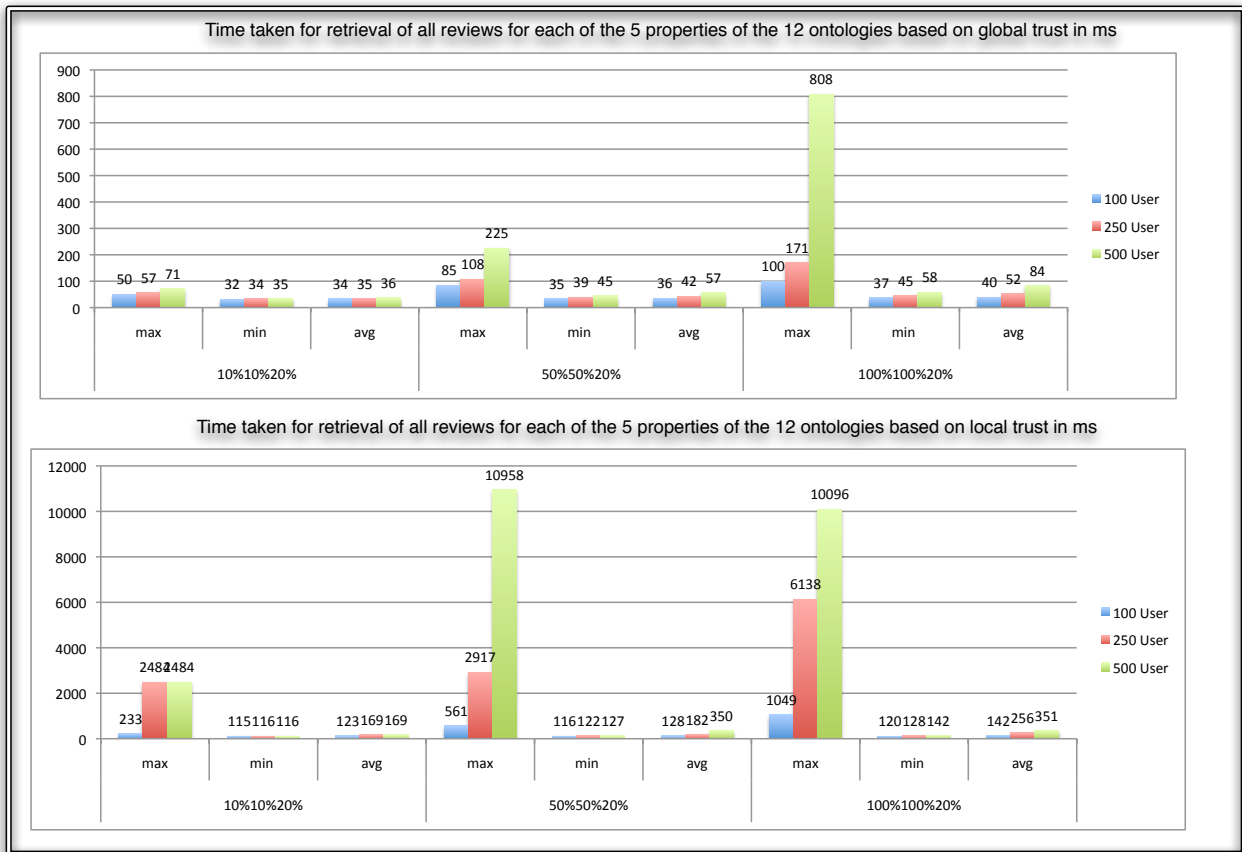


Figure 2.9: This graphic presents the times taken (min, max and avg) for returning all reviews for all 5 properties of all 12 ontologies in ms once based on global and once based on local trust. 50 runs were performed. (Run on MacBook Pro)

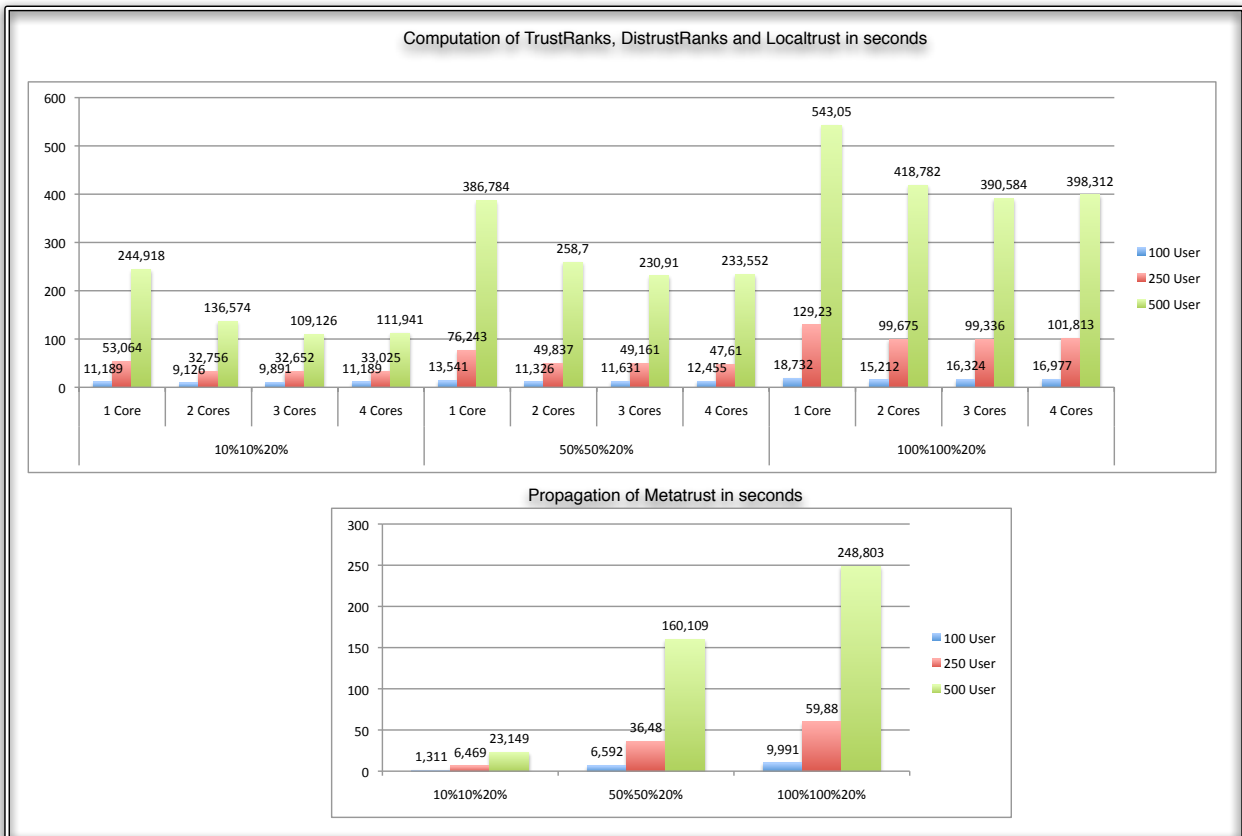


Figure 2.10: This graphic depicts the results of the benchmark of the computation and meta trust propagation. Data is presented as time in seconds. (Run on Intel Core 2 Quad)

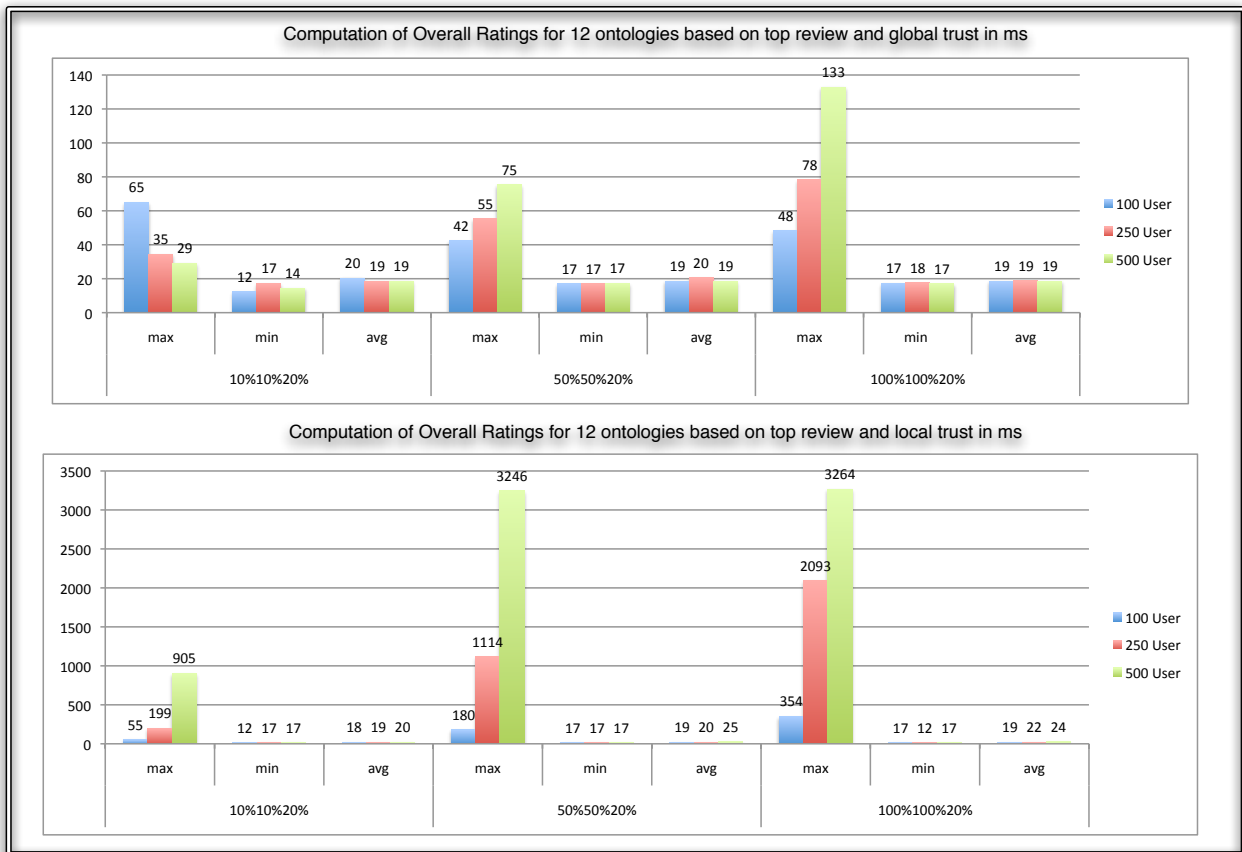


Figure 2.11: This graphic presents the times taken (min, max and avg) for returning the overall rating for all 12 ontologies in ms once for global and once for local trust. 500 runs were performed and only the top review was considered for the computation. (Run on Intel Core 2 Quad)

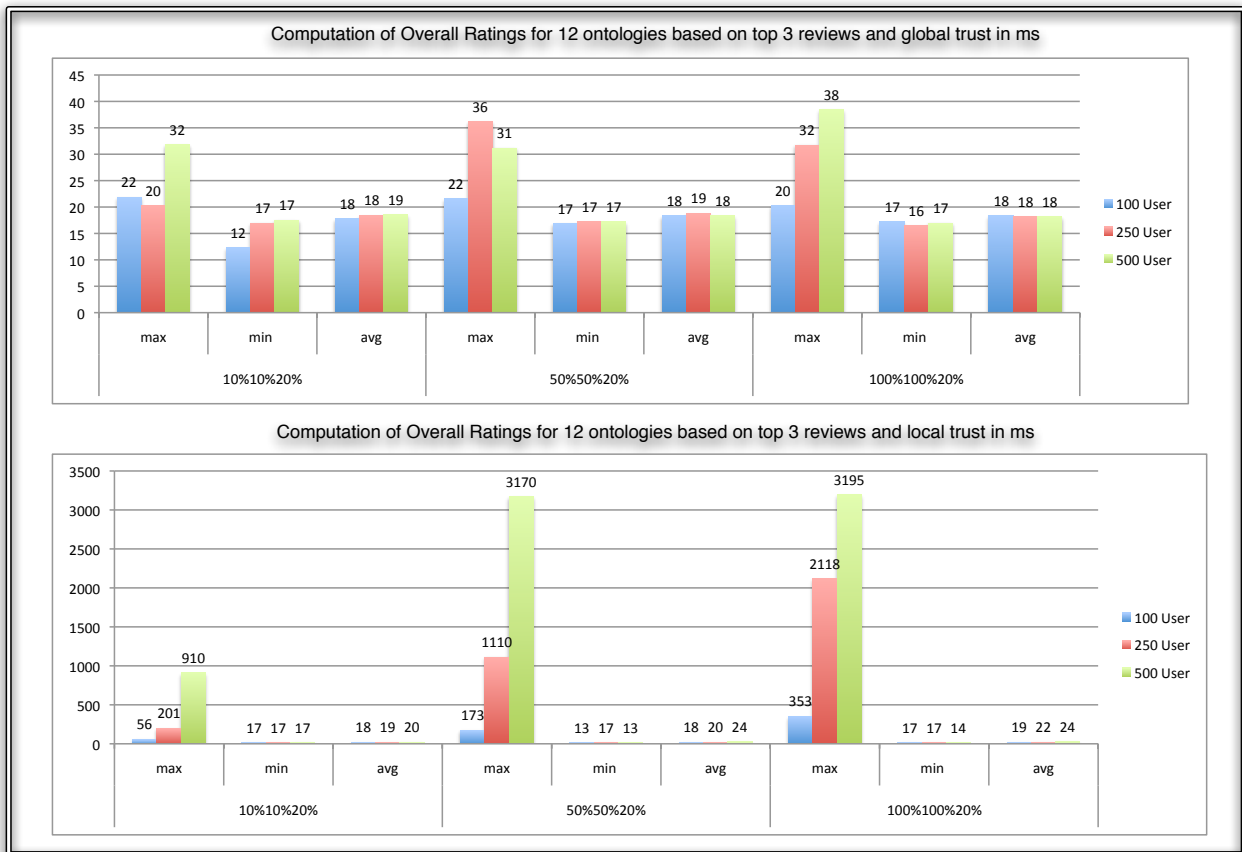


Figure 2.12: This graphic presents the times taken (min, max and avg) for returning the overall rating for all 12 ontologies in ms once for global and once for local trust. 500 runs were performed and only the top 3 reviews was considered for the computation. (Run on Intel Core 2 Quad)



Figure 2.13: 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. 500 runs were performed and all reviews were considered for the computation. (Run on Intel Core 2 Quad)

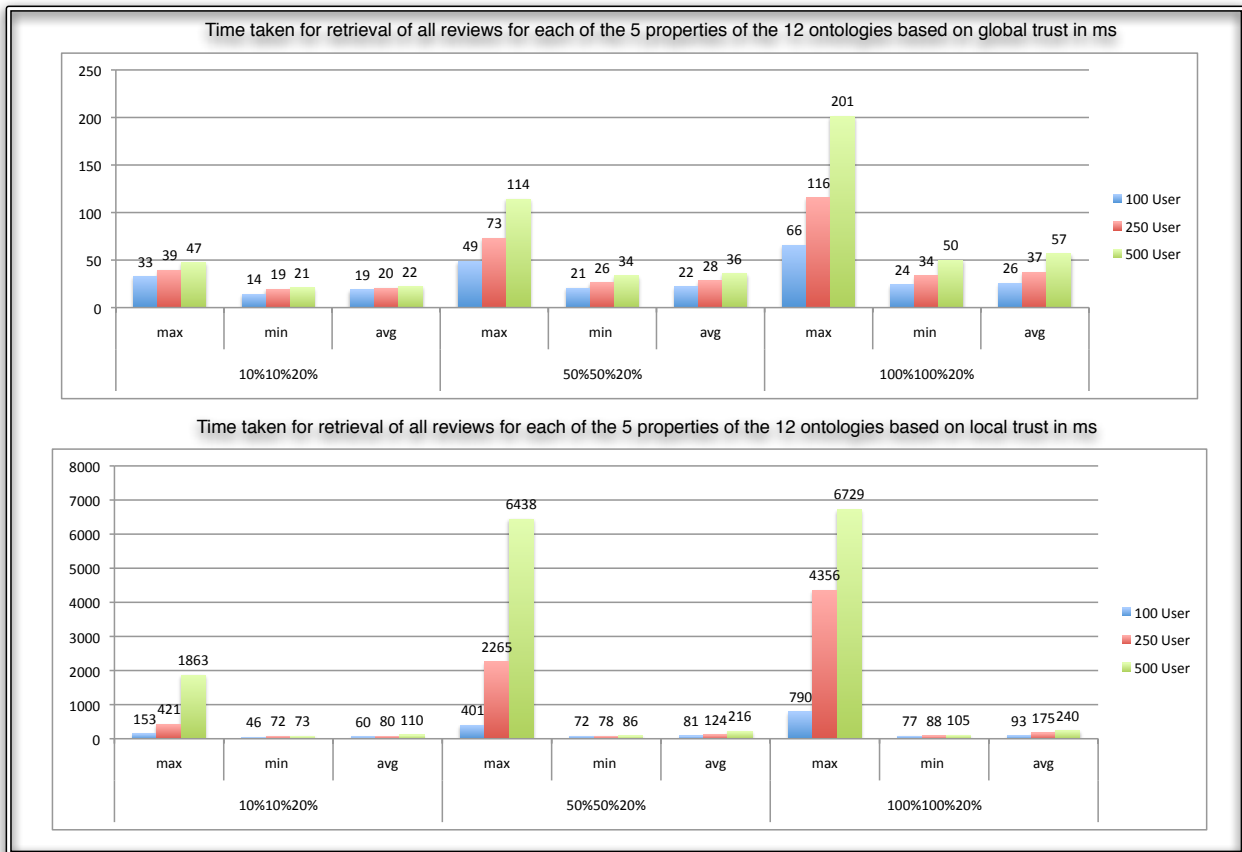


Figure 2.14: This graphic presents the times taken (min, max and avg) for returning all reviews for all 5 properties of all 12 ontologies in ms once based on global and once based on local trust. 50 runs were performed. (Run on Intel Core 2 Quad)

2.5 Result Analysis

We will now analyze the improved results of the benchmark and try to explain how this speedup was realized. The general analysis provided in the last deliverable still holds. We will start with Figures 2.5 and 2.10, which depict the computation of trust and the propagation of meta trust.

2.5.1 Meta-trust Propagation and Trust Computation

As can be seen better in Figure 2.15, which provides a direct comparison between both the old code and the new code on the two test computers, the performance for meta-trust propagation has increased on average by a factor of 10 (over all settings and only comparing the results on the same machine). The same is true for the performance of the trust computation. It can be seen that main memory is a limiting factor in the computation when looking at the comparison for the new code on the Core 2 Duo and Core 2 Quad. While for smaller scenario they perform more or less equally fast, for the scenario with more data (e.g. 100%100%20% with 500 users), the difference between the two computers increases. This is because on the smaller machine with only 3 GB of Ram, not everything can be kept in main memory, and is stored on disk (Swap file). When this happens, the performance decreases, since disk access is much slower than main memory access.

Leaving memory related variations in the execution time aside, it seems that the duration for meta-trust propagation grows linearly with the increased amount of data, and roughly squared with respect to the number of users (also taking into account that the amount of data grows with the number of users). For example: The time taken for 100 users in the 100%100%20% setting on the Core 2 Quad is 10 seconds, for 5 times as many users it is 250 seconds ($= 10 * 5^2$). The time for 500 users in the 10%10%20% setting is 23 seconds compared to the 250 seconds in the 100%100%20% setting. It still has to be noted that 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 meta trust statements. If we only increase the number of users, but not the number of reviews, the computation takes roughly the same time. For example, the propagation of meta trust for the setting 500 users with a 2%, 2%, 4% setting (which is equivalent to the 100 User 10%10%20% setting) takes 4.5 seconds instead of 4.9 seconds for the case of 100 Users (which indicates that the number of users alone does not affect the duration of the meta-trust propagation, mainly the amount of data to process is decisive). The results for this comparison can be seen in Figure 2.16.

The speedup for the computation was mainly due to the improved database interaction. The database access is sort of the lower barrier for the execution time which cannot be lowered any further. Looking at Figure 2.17 it can be seen how the distribution of time changes when the computation is parallelized. Since database access is not parallelized, this time is solely depending on the amount of data inserted and read and stays the same even if more threads are used for the computation. The computation itself sees improvement, but here the overhead for parallelization also has to be taken into consideration, which explains why the time taken for computation does decrease linearly with an increasing number of threads used. For most realistic scenarios, it seems at least 2 and at most $n-1$ threads should be started on a n -core processor (with $n \geq 3$.) 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 with the new code, even in the worst case setting, the computation time is around 6 seconds per ontology-property combination. In a more realistic setting (few reviews and trust statements) this time is now less than 1 second.

2.5.2 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

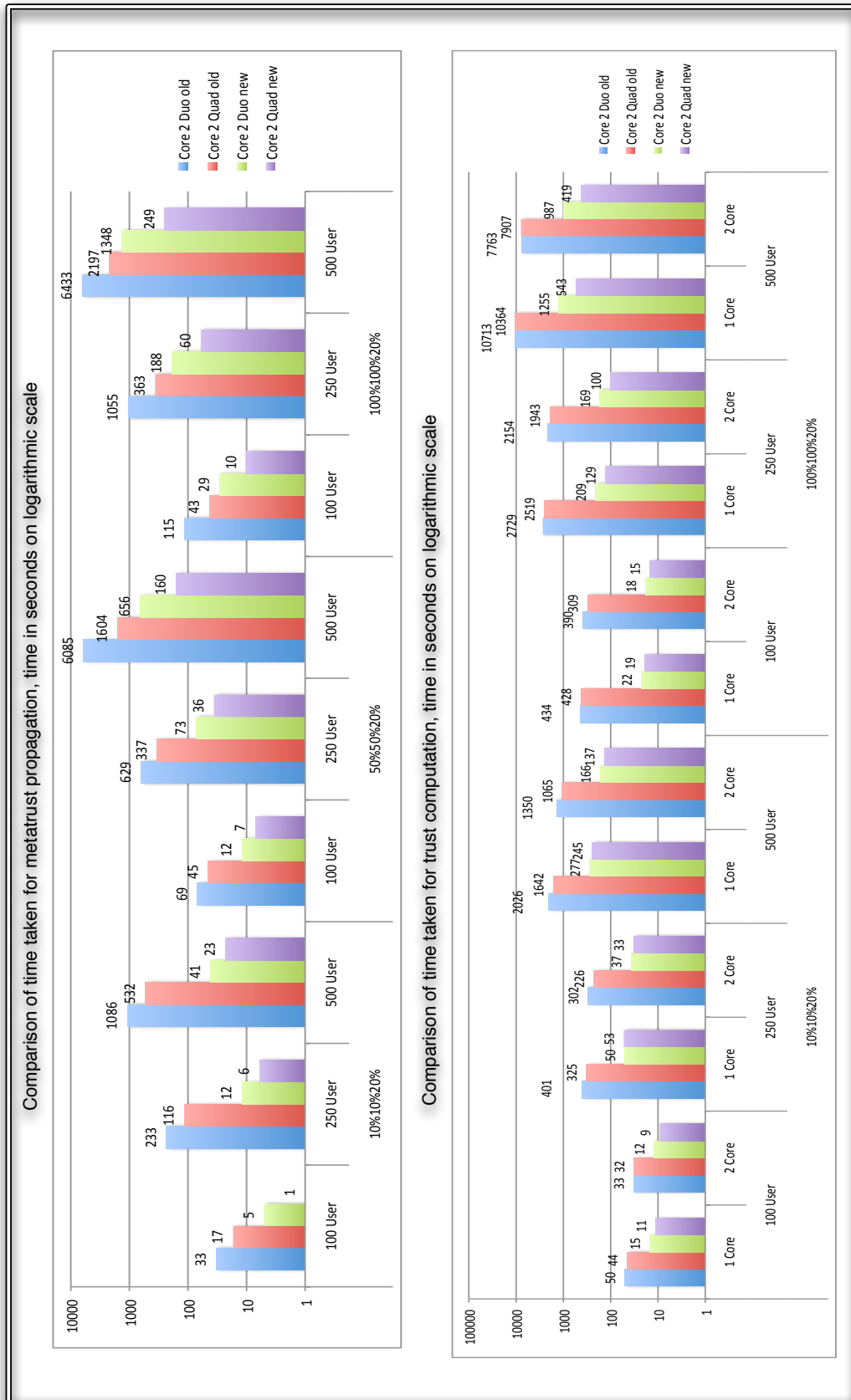


Figure 2.15: This graphic presents a direct comparison between the two versions of the code and the two different computers. Please note that the left axis has a logarithmic scale.

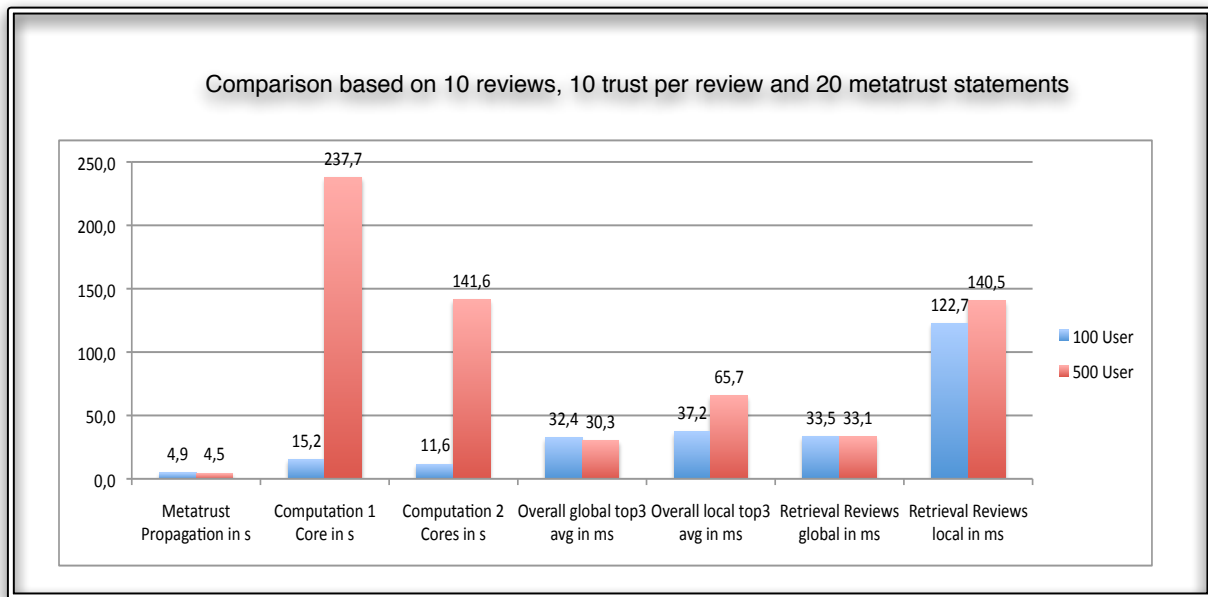


Figure 2.16: This graphic presents a comparison between the time taken for different actions based on the same review and trust statements but increased number of users. (Run on MacBook Pro)

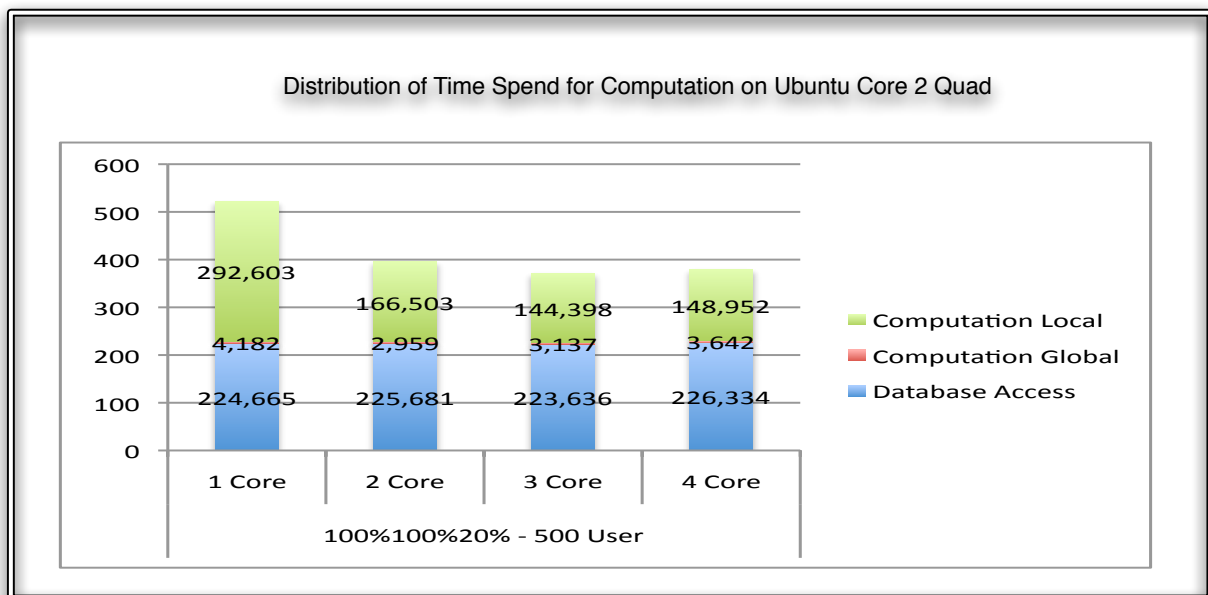


Figure 2.17: This graphic presents the distribution of time spend for the different tasks during the trust computation for a different number of threads.

considerations). For each of these settings, we base the computation on both local trust and global trust. Furthermore, we run each computation 500 times, to get more accurate average results. We measure the execution time for each of the 500 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 using the caching method (see section 2.2.2 above). 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 2.6, 2.7, 2.8, 2.11, 2.12 and 2.13.

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. With the new code, also the average retrieval time for global and local trust seems to be the roughly the same. We have compared the results of the old and the new code for in Figure 2.18. Please note that we have chosen to compare the minimum times rather than the average times, because we have used a different number of runs during the two performance benchmarks, and the only comparable time is the minimum time. Also note that the more number of runs you use, the closer the average time gets to the minimum time. The maximum time is mainly encountered when there is no cached information (cache miss). For our best machine, the average minimum time to retrieve both local and global based results now is 17 ms for all 12 ontologies. That means that we have improved computational performance to roughly 1.5 ms in the best case.

2.5.3 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 global trust. We have benchmarked retrieving all of the reviews in aforementioned personalized order for all ontology-property combinations. In Figures 2.9 and 2.14, you can see the results both based on local trust and global trust. We have run each task 50 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 around 0.5 for each ontology–properties based on global trust (depending on number of users and thus reviews). Retrieving the results based on local trust takes about 1 ms on the fast machine.

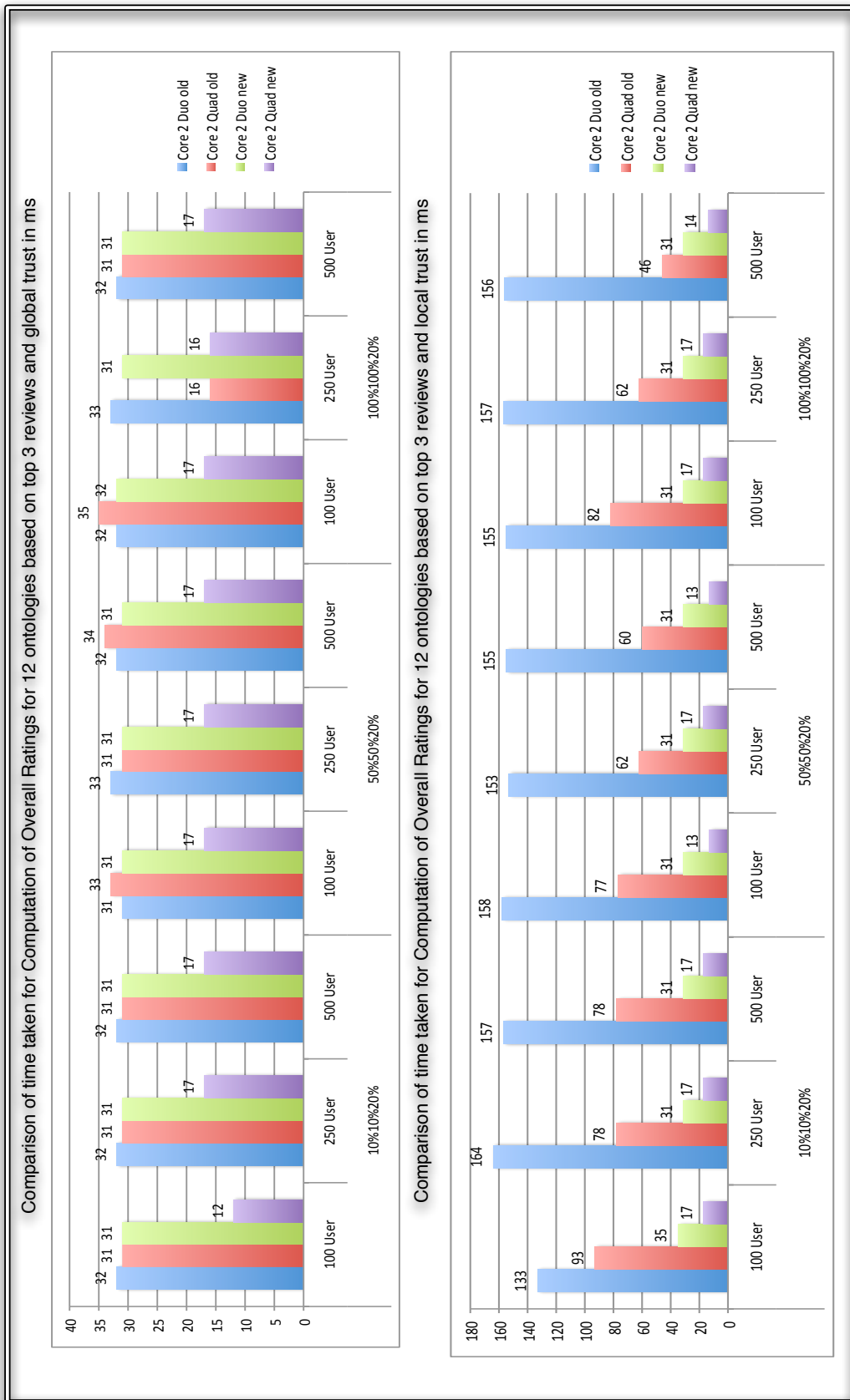


Figure 2.18: This graphic presents a direct comparison between the two versions of the code and the two different computers. Please note that only min times are compared and the timer is in ms.

2.6 Results Learnt from Benchmark

Since many operations rely on a fast database, having good hard disks 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, the caching method can be called to cache results, for example for logged-in users. This leads to fast runtime response times.

It also seems to be 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. Also sufficient memory and ideally a 64-bit system with 64-bit java improve the performance. Depending on how important it is to take the latest user data into account, the frequency of overall re-computation can be increased or lowered. Amazon.com, for example, take 24 hours to take a trusts statement into account. In case a really fast re computation 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 for all ontology–property combinations, in the most extreme case, you could use one machine per combination and later merge the results.

The runtime performance looks even more promising after the code optimization, and should not lead to bottlenecks at runtime.

Chapter 3

Simulation-based Evaluation of the Topic-Specific Open Rating System

3.1 Introduction

After testing the performance of the system in the prior chapter, we wanted to know whether the algorithms work as expected. One way of evaluating the behavior of a system is to run a simulation. Using synthetic data to test the system behavior has several advantages over purely analyzing real world data. Whilst it is important to see whether the algorithms in the system hold for real world data as well, it is difficult to check the behavior in edge cases. Also there is no control over user behavior in real world test data. Therefore we decided to run a simulation for our Topic-Specific Open Rating System (TS-ORS) in addition to performance tests. The main purpose of the simulation is to check the system behavior in a controlled environment and draw conclusions for its use in real world systems. In order to achieve this, we have run the same task (computing overall ratings for ontologies based on trust in the system) several times, altering parameters in order to see how different coverage or errors made by the users affects the outcome. In the remainder of this report, the system we use as a testbed is introduced, as well as our evaluation setup. Later the results of the different test runs are shown and analyzed. We finish with a conclusion.

3.2 Setup

This section will provide a quick overview of the system employed to run the evaluations and the setup we created.

3.2.1 System

We use the code base that was integrated into Cupboard [dL09] for our simulation. Because of the specific application setting, we only tested topic-specific trust for ontology properties, not for domains (since at the current moment ontology domains are not used in the system). Nevertheless, the results shown in this report based on property-specific trust can be generalized for topic-specific trust covering properties and domains. As is described in [LSNM06, SAd⁺07b], the main feature of the TS-ORS is to rank reviews and based on the reviews also the ontologies according to computed ratings for the ontology derived from reviews and trust information available to the system. Based on the information on user-to-user trust, the TS-ORS computes both global (not user specific) and local (user specific) trust rankings. The global trust information can be used for the users not connected to the web of trust, or not identifiable (not logged in). The web of trust (WOT) can be seen as a network where users are nodes and edges are trust statements. Global trust is not user-specific, unlike local trust, which features personalized trust information on the other users. Based on knowing who the most trusted user is (note that this can be user-specific), the system can use the star rating of the review of that user to compute an overall rating of the ontologies (again based on parameters the users

can define). The main purpose of the evaluation described in this report is to find out whether the algorithms work as intended, given different scenarios that could be encountered as such in real world systems.

3.2.2 Basic Experiment Setup

While it would have been possible to run the experiment with a very low number of users and ontologies, we chose to have a larger number of instances and compute the average over the results, since we felt this could even out too random results. The data we generated is the user-base, the ontologies and reviews for the ontologies, and finally the trust a user places in another user.

Users

The simulated users in our case are divided into three groups, which we call *good users*, *controversial users* and *bad users*. The main motivation here is rather showing how different subgroups affect the system, the labeling should not be taken literally. We could as well have called them group A, B, and C. The good users do not try to game the system (by promoting SPAM or bad ontologies), they accurately review good ontologies and bad ontologies. The controversial users represent a subgroup that is not trying to game the system but has a taste (e.g. special needs) different from the mainstream. They like controversial ontologies more than the good ones, but also do not like the bad ones. The bad users are the ones trying to game the system, they rate bad ontologies highly and try to form subgroups to boost the popularity of their infiltrated bad content.

The rationale behind choosing this setting was to mirror situations that occur in real life. These systems have a large number of users that use the system as intended, but not all users have the same taste or needs. In order to mimic the “taste” difference, we have the two groups good users and controversial users. In order to mimic malicious users, (like SPAM-bots) we have the group bad users.

For the experiments, we have created 100 users total, 60% of them are good users, 20% controversial users and 20% bad users. It has to be noted that this decision does influence the outcome, since for some measures, the largest group will automatically dominate the rankings. This does not hold true for the local trust measure. We will mention this problem several times in the remainder of the document. We chose this contribution, because we felt that in a normal, well maintained system most of the users would fall in our good users group. However, it is not important for the conclusion how the distribution was made, since the findings can be generalized for different distributions.

Ontologies

With regard to the ontologies, we distinguish good ontologies, controversial ontologies and bad ontologies. Again we could have called the three groups A, B, and C, we are aware that most ontologies out in the real world would fall into the controversial and bad category, and few would be universally agreed to be good. Nevertheless, the main point here was to have different sets of ontologies and see whether the right set of ontologies could be returned to the users looking for it. The good ontologies represent overall good quality; they are also liked by the controversial users, however not as much as the controversial ontologies. The bad ontologies are of bad quality and are only promoted by bad users. Each ontology has 5 properties which are rated.

We have created 50 ontologies of which 50% were good, 20% controversial and 30% bad. The distribution of good, controversial and bad ontologies does not influence the outcome of the experiment too much, since all the trust computations were made independently. It is true, however, that for the small coverage cases where we randomly chose which ontology to review, the data density in terms of ratings for the ontologies would be bigger in a larger group.

Reviews

For each of the different user groups, we have defined rules, according to which the reviews are generated for each ontology and property:

- “Good” users like good ontologies, and rate them highly (5 star). They differ over the controversial ontologies, 50% like them (4 star), 50% do not like them (2 star). They all dislike bad ontologies and rate them 1 star.
- “Controversial” users like the good ontologies (4 star), but like the controversial ontologies better (5 star). They also dislike the bad ontologies (1 star).
- “Bad” users dislike (1 star) all ontologies except for the bad ontologies (5 star).

Trust

In terms of trust, we have decided that each user trusts only peers in his group and distrusts the rest, that means a good user trusts all good users, but distrusts controversial users and bad users. Bad users only trust bad users, reflecting the attempt to build a network to boost their importance.

3.2.3 Variations During Test-Runs and Alteration of the Setup

We have created three different scenarios, each consisting of 12 test-runs, resulting in 36 runs total.

Scenarios

- The first scenario is reflecting a setting similar to what Guha et al [GKRT04] used for their experiment. Trust is assigned globally, not topic-specific (that means, that it is only possible to address an overall trust in another user, not trust in certain abilities). In this setup, we have also constructed the dataset under the assumption that all users were able to rate the 5 different properties of the ontology equally well. Therefore, the ratings and the trust is assigned as described in the section above.
- The second scenario tries to discover what happens if users have different reviewing specialties, i.e. are only experts in one area. To mimic that, we have distinguished their reviews in good reviews, i.e. reviews in their area of expertise, and bad reviews, i.e. reviews in the area where they are not experts. Specifically, if a user has for example expertise for rating property 1, these reviews would be good and according to the schema outlined in section 3.2.2 above. For all other properties, the review would be marked bad and the original rating was inverted (i.e. 5 became 1, 2 became 4 and 3 stayed 3). For the bad users this distinction was not made, since the assumption was that they are not interested in providing good reviews, but just promoting their bad ontologies. Therefore their rating would remain the same (1 star for every ontology except for bad ontologies, which are rated 5 stars). Trust, however, is still assigned globally within the group, i.e. the good users trust the good users, but only 20% of the trusted reviews are actually good. In the system described by Guha et al [Guh03, GKRT04], only global trust is allowed.
- The third scenario has the same review setting as the second scenario (i.e. expertise only in certain areas), but allows for topic-specific trust. Here, the users are only trusted by their peers for the properties for which they provide good reviews, and not for the other ones. The exception is once again the bad user group, in which still all users trust each other globally. The idea is to test how the user can benefit from the ability to assign topic-specific trust against having to use global trust.

Sparsity of Data

One of the problems in real-world systems also encountered by many recommender systems is data sparsity [HKTR04]. If only a few users review ontologies and state their trust towards other users, the algorithms have to function on as well. In order to see how the algorithms can handle data sparsity, we have run each setting with a 100%, 50%, 10% and 5% coverage. That means that for the 100% case each user reviews all properties of all ontologies and states his trust for all other users. In the 10% setting each user reviews a randomly selected subset of ontologies, exactly 10% of all ontologies in the system. Also each user now only states his trust towards 10% of the other users (again, the group of users was selected randomly). For simplicity, if an ontology was part of the randomly chosen subset, all properties were reviewed by the user. So in the most sparse setting (5%), given our experiment size of 100 users and 50 ontologies, each user reviews around 2 ontologies and assigns trust towards 5 other users.

Errors in Judgement

Another important aspect is that users do not always act 100% as expected, but can make mistakes like not identifying a good user or trusting a bad user. In order to measure what effect this has on stating trust statements, which in the end are used for the computation of the trust matrices, and thus the entire ranking process, we have run each of the sparsity settings once without error, once with 10% error and once with 20% error. An error would mean that the opposite of the normal action is performed, for example a good user would trust a bad user or might distrust a good user. A 10% error would mean that out of all the trust statements given by a user, 10% are wrong, i.e. complimentary to the strategy defined in section 3.2.2. This holds for all user groups, it is assumed that also a bad user might make mistakes.

3.2.4 Remarks on Implementation

Since we have run the tests on the identical code-base as used for the Cupboard system, the trust computation was executed for each ontology–property combination separately. The global trust statements were modeled as meta trust statements, i.e. statements of trust or distrust between two users, which are then broken down to individual trust statements on the ontology–property level by the system. So if, for example, user A trusts user B globally, the system searches for all reviews by user B and assigns a trust statement to them. For the third scenario, a property-based meta trust statement was used, i.e. the trust is only propagated to reviews for that property.

3.2.5 Result Generation

As mentioned before, the 3 scenarios with 4 sparsity settings and 3 error settings yielded in 36 test runs. For each run, we computed the global und local trust for each ontology–property combination and then retrieved several ratings. For each of the hereafter mentioned runs, we retrieved the rating for each ontology in the system and averaged it with the results of the other ontologies of that category. Since the computations require some parameters, we have kept these stable throughout the whole experiment. They are $\alpha = 0.7$ (used to combine TrustRank and DistrustRank) and $\nu = 0.8$ (used to weigh the top N reviews using descending importance, see equation 3.1). The 5 properties were weighted evenly. In the case of local trust (i.e. trust that is user-specific), we have received the rating for each user and averaged it over all users in the group. In the results we distinguish the three different ontology types good, controversial and bad.

$$w_i = \frac{(\nu)^i}{\sum_{i=1}^N \nu^i} \text{ with } w_i \text{ weight for } i\text{-th review, } 0 \leq \nu \leq 1 \quad (3.1)$$

- First of all, we computed the average rating. This metric is the easiest metric obtainable and does not require any trust computation in the background.

- Then we computed the average rating based on all reviews and global trust (this means, that also bad reviews will be taken into account, but according to their position in the result set, the impact is minimal because of the decrease of importance according to position (see Equation 3.1).
- After that we compute the average rating based on the top 3 reviews and global trust (this means that only the first 3 reviews according to global trust measures will be taken into account).
- Finally we compute all the local trust combinations, i.e. for each of the three ontology types, we measure the average rating for each of the three user types, yielding in 9 combinations. We compute this measure once based on the top 3 reviews, and once based on the top review, all based on local trust for each individual user in the group. That means that for the case of good ontologies and good users, we retrieve for each of the 60 good users the rating for each of the 25 good ontologies. Then we compute the average.

3.3 Results

In this section we will discuss and analyze the outcome of the simulation.

3.3.1 Expected Results

Since we have clearly defined rules how the three user groups act, we would imagine the rating algorithms to also retrieve ratings according to these rules in the case of local trust. That means that if all controversial users rate controversial ontologies 5 star, this is supposed to be the rating returned by the system for controversial ontologies and controversial users when only users in the same group are trusted. It is clear that for the global measures the size of the groups matters, and the question of whether the groups are confined or whether there are links between the groups. That means that in our case, since the good users group is the largest (60%), and there are no links between the groups, we expect their opinion to dominate the ratings based on global trust. For the local trust, it is important that the trust is accurately assigned. If for example a bad user is trusted by a good user, this will also connect the good user to the bad reviews, and will influence the computed rating for the ontologies. We expect to see this effect when the element of chance is introduced to simulate erroneous behavior. We furthermore assume the results to be more accurate the more trust statements and rating coverage we have. This is due to the fact that the local trust based ranking has to rely on global trust values for the users, for which no trust information can be computed by means of local trust (i.e. they are not connected in the WOT). If a bad users is not connected to other bad users for example, the system has to refer to global trust ranking of users, leading to a result different from what was expected. We furthermore expect the topic-specific trust approach to be more accurate in the case that reviewers have different reviewing skills for different properties of the ontologies (comparison between scenario 2 and 3). Lastly, we will analyze the effect of data sparsity on the rating results in section 3.10.

3.3.2 Scenario 1

As mentioned before, each scenario consists of 12 runs total, 4 different data density settings, and 3 different error settings. In the figures you will see the results for the different density settings side-by-side in one diagram. We have one figure for each error setting.

0% Error

The 0% error setting is the one that is supposed to work as described in section 3.3.1. If we look at the results shown in Figure 3.1, it is evident that given perfect coverage, the algorithms indeed produce the results expected. The average rating for controversial ontologies and controversial users is 5 stars, the same holds true for good users and good ontologies. Furthermore, the bad ontologies are rated 1 star for both good

and controversial ontologies. As coverage goes down, the results move towards majority taste. This is due to the size of the test data we have used and will be explained further in section 3.10. Basically the problem is that a user is not always connected to the WOT and when he is not connected, the global trust metric has to be consulted for trust information. In these sparse density cases the top1 based rating produces better results than the top3 setting. This is due to the fact that if there are less than 3 locally trusted reviews, globally trusted reviews are factored into the equation. As a general observation, global trust outperforms the simple average and local trust outperforms global trust (with top1 outperforming top3), with outperforming defined as being closer to our expected results.

10% Error

The results for our 10% error setting can be found in Figure 3.2. One main observation is that once the boundaries of the peer group are violated by trusting a user from another peer group and furthermore also distrusting some of your peers, the largest group becomes predominant. The good users group is the biggest one in our experiment, therefore dominating the results of the controversial and bad users. Unlike the setting with no errors, these groups cannot retrieve a review from their peers as top review, since once one user from the group trusts another user from the good user group, this user will receive enough trust to become the predominant reviewer. So in this setting the expected results of 5 for controversial users and controversial ontologies is not encountered, neither the rating of 5 for bad ontologies and bad users. The difference in coverage can be explained as in the other scenario without error, namely a lack of connectedness to the WOT.

20% Error

In a setting with 20% error (see figure 3.3), the results resemble very much the results of the 10% setting. The effects described above are just more visible. This also allows for the conclusion that within a certain range of error, the influence on the result is minimal. That means that there is not much difference between making errors in 10% of the cases or 20%.

3.3.3 Scenario 2

To see which drawbacks allowing only global trust statements has on the ranking results, in the second scenario we assume that each user only has one area of expertise (in our setting one ontology property), which he can review properly. The rest is also reviewed, but the reviews are of bad quality (to mirror that, we inverted the correct rating for the ontology-user combination. In order to at least get good results for the properties that these users can review, they are trusted globally (remember, that there is no more fine-grained way to assign trust in this scenario). The alternative of not trusting the user at all (because after all 80% of the results are bad) is not an alternative, since this way nobody could be trusted and the rating would have to completely rely on global trust, which would then favor the bad users who trust each other.

0% Error

In this setting we expect to see incorrect ratings to be computed due to the fact that the global trust statements assigned based on the good reviews also cover all the bad reviews of that user. As can be seen in Figure 3.4, the results are the complete opposite of what would be desired. The good ontologies rank lowest and the bad ontologies first. So the expectations were fulfilled, and the fact that in the end 80% of the reviews were incorrect (only 1 out of 5 properties could be reviewed correctly) led to the expected bad results. Still, the same behavior as described in scenario 1 is encountered here. The lower the coverage, the worse the connectedness to the WOT, and so the average rating for bad ontologies for bad users drops from 5 to roundabout 4 (now dominated by the rating of the good users. Please remember that the bad users were still expected to review maliciously for all ontologies, which explains the 5 star given sufficient coverage.

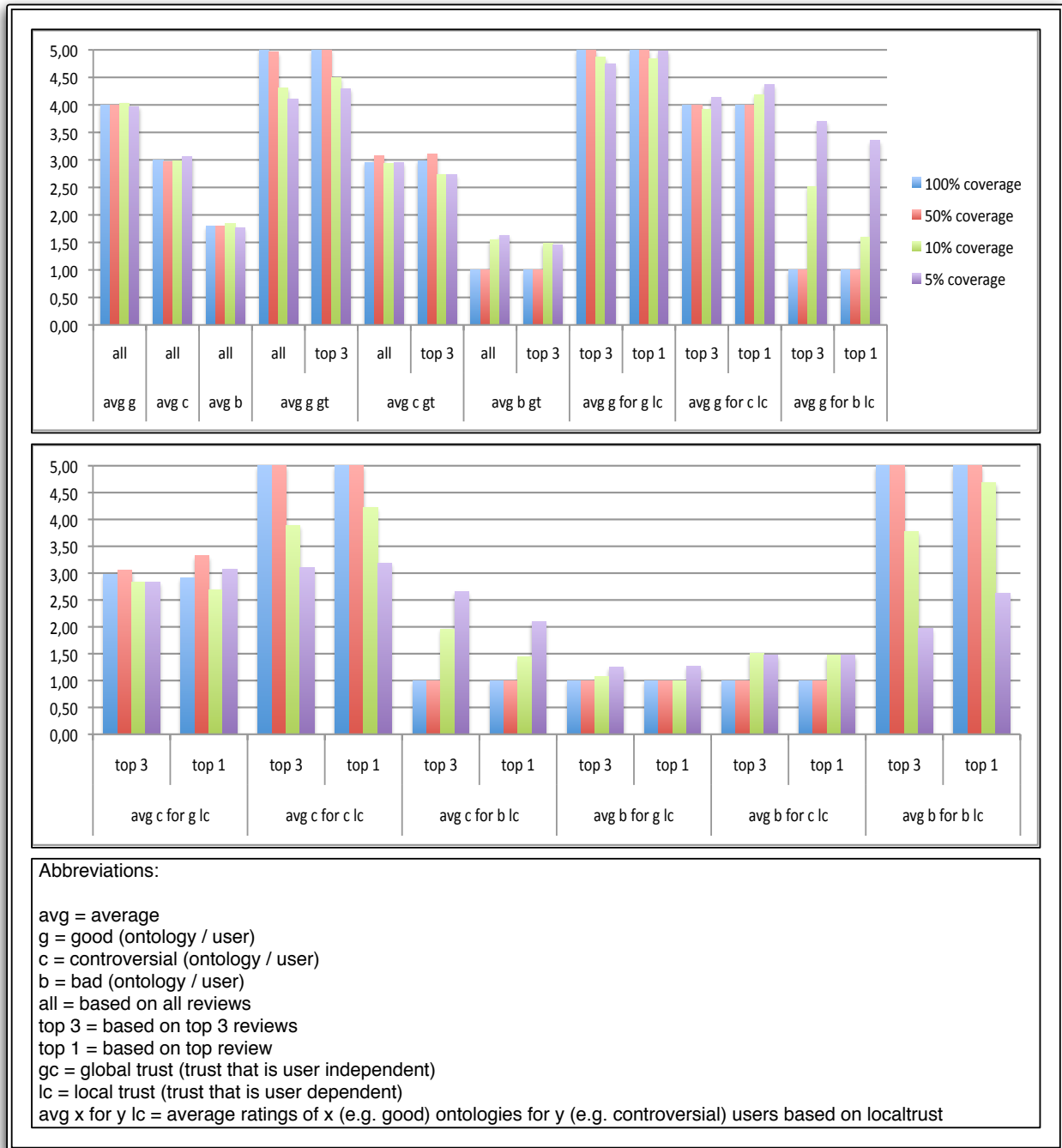


Figure 3.1: First Scenario with 0% Error

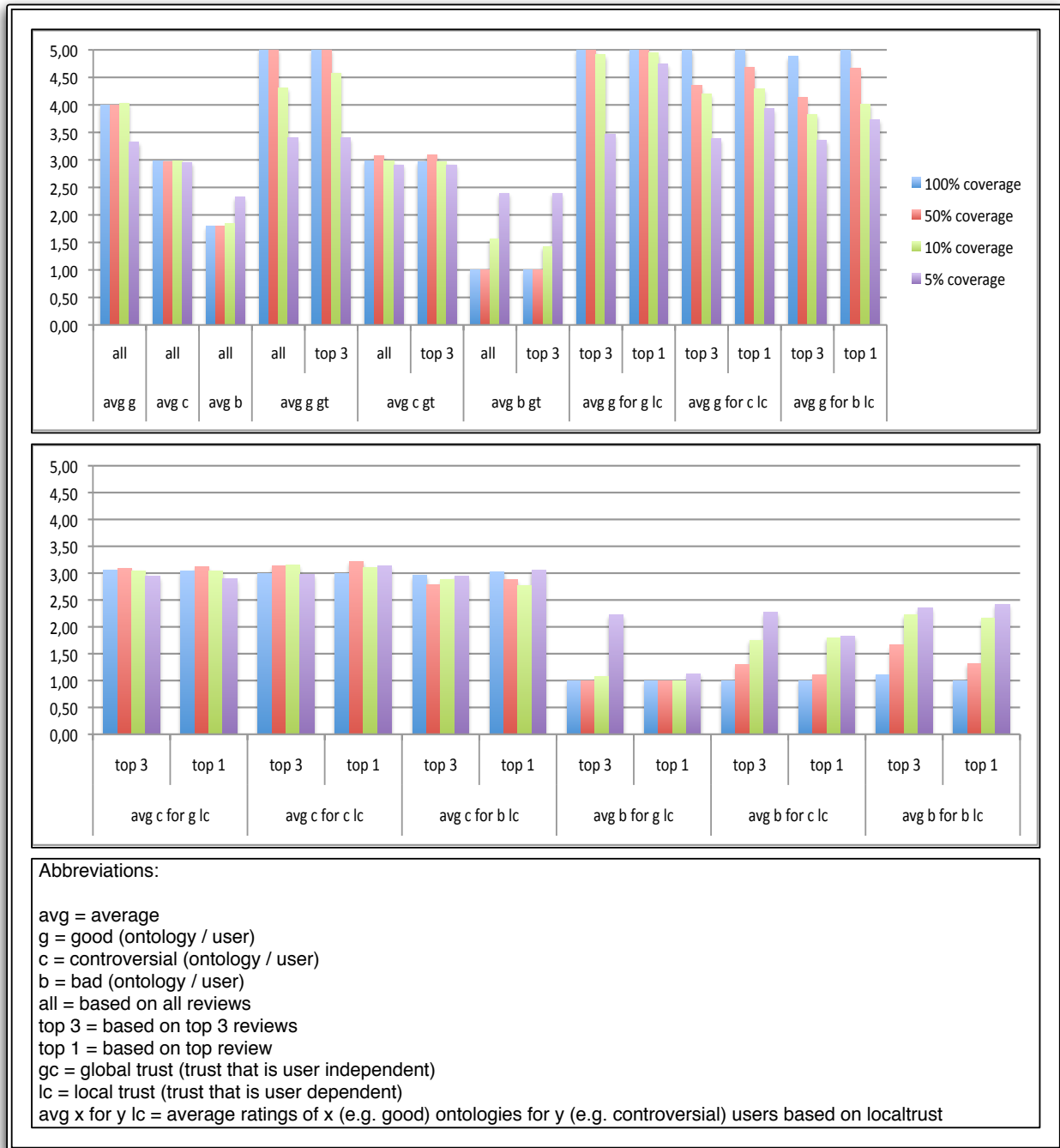


Figure 3.2: First Scenario with 10% Error

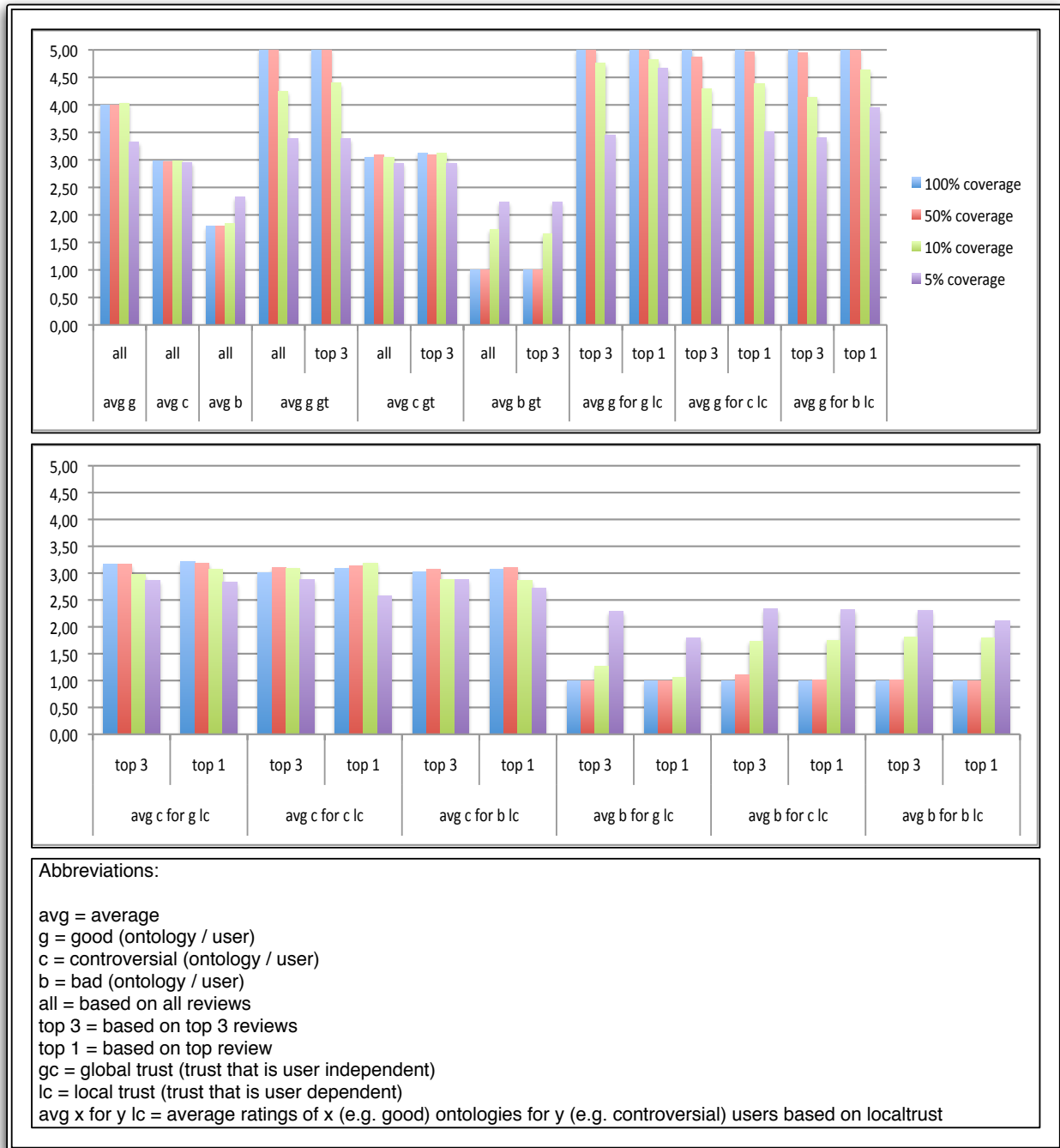


Figure 3.3: First Scenario with 20% Error

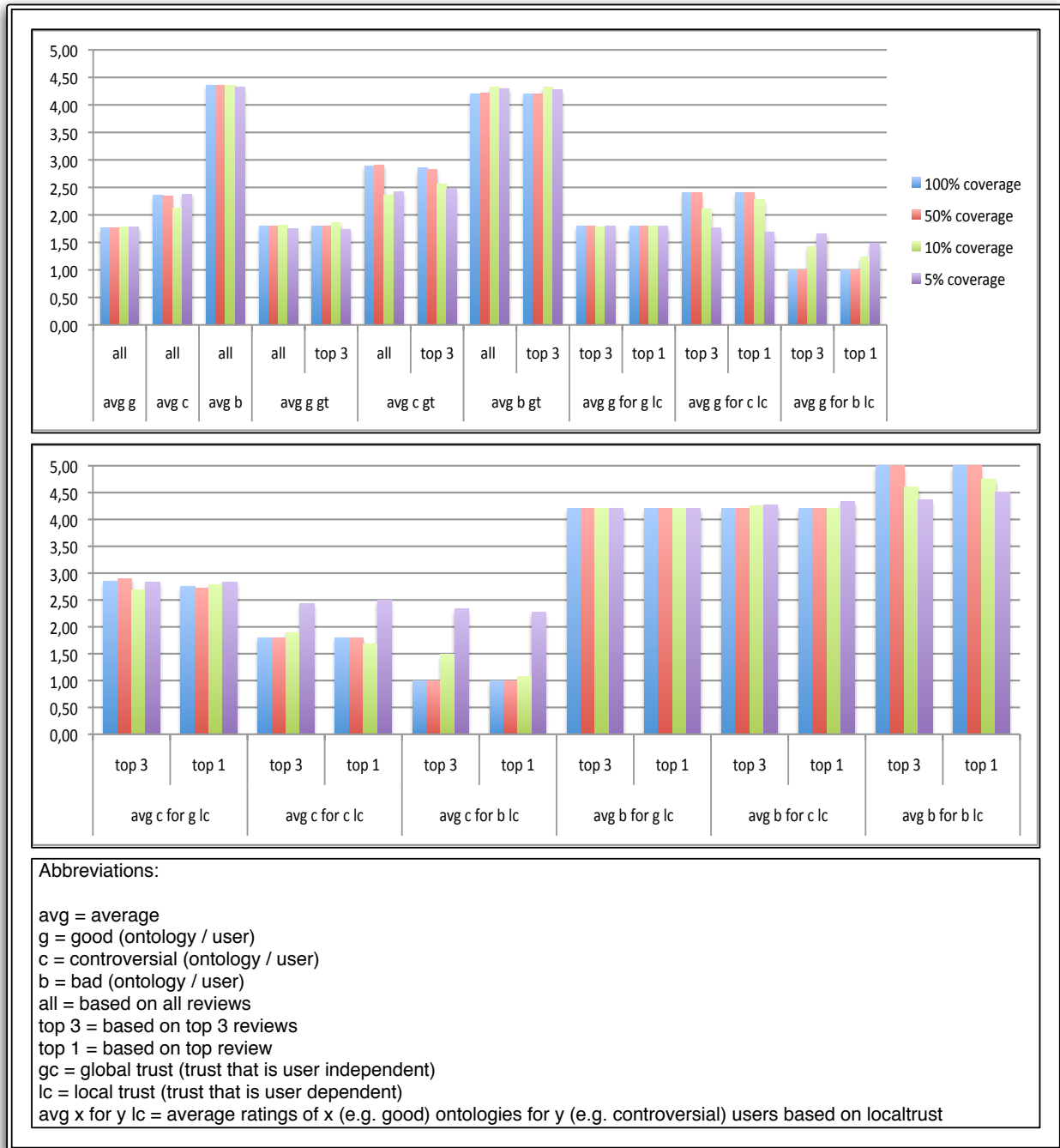


Figure 3.4: Second Scenario with 0% Error

10% and 20% Error

Here the same observations can be made as in scenario 1, i.e. the predominant user group affects the ratings of the other subgroups. The results can be found in Figures 3.5 and 3.6.

3.3.4 Scenario 3

Since the claim is that topic-specific trust can overcome the drawbacks of global trust, i.e. having to trust users also for bad reviews when trusting the good ones, we expect that the outcomes will be similar to scenario 1. In this setup the users are only trusted for the properties for which they can provide good reviews, and not distrusted otherwise.

0% Error

As is evident from Figure 3.7, the topic-specific trust indeed does provide the expected ratings given enough connectivity to the WOT. Even though the user reviews are the same as in scenario 2 (as can be seen from the overall average ratings), the ratings for local trust-based and global trust-based results are as good as in scenario 1. So even though there are 80% wrong reviews in the system, the algorithms still produce the desired outcome, based on accurate trust assignment and sufficient connectivity. With decreasing coverage we can once again see how the expected results blend in with the global trust-based results for the respective coverage.

10% and 20% Error

Also in scenario 3 we can find the described effect of the predominant group influencing the rating results of the other subgroups (see Figures 3.8 and 3.9). Still it is noteworthy that even with 20% of all trust statements being wrong, the average rating is outperformed. In other words, even if not all trust information is correct, it is better to employ trust-based algorithms for ranking than relying on basic arithmetic measures like average.

3.3.5 Comparison 5% Coverage with 100 and 1000 Users, 0% Error

As indicated before, we expected the worse performance of the algorithms at smaller coverage levels to be due to the missing connectedness of users to the WOT. After all, the 5% coverage equals to two reviews and 5 trust statements in the 100 user setting. We have compared the influence of increasing the user-base while keeping the number of ontologies and level of coverage equal. The results can be found in Figure 3.10. We furthermore analyzed the setting with 1000 users in more detail, to find out how the derivations from the expected outcome can occur. As discussed in section 3.3.6, the theoretical lower boundary for a system that delivers expected results for our setup lies with 3 reviews per ontology (one for each group), and one trusted peer user for each ontology per user. That amounts to 3% coverage in reviews and 1% coverage in trust per ontology (in a 100 user setting). Since we have employed randomization for the simulation, the reviews and trust were not assigned optimally, but randomly based on the rules explained above. This could lead to settings where users are not connected to the WOT, and therefore for them, no local trust information can be used to compute the overall rating. In order to see how many users are disconnected for each ontology (since for all these users, in our experiment setup their trust results would contain global trust information), we ran an analysis on the setting. For the setup 100 users, 5% coverage and 0% error, the results can be found in Figure 3.11. The graphics displays, for how many ontologies the number of users disconnected from the WOT falls into the percentage of users displayed on the x-axis. As is evident from Figure 3.11, for several ontologies a larger percentage of users was not connected to the WOT, in some cases even more than 90%. Therefore it is not surprising that the average ratings based on local trust deviates towards the ratings based on global trust, since in some cases, the ranking algorithm has to rely on global trust. In the rest of the cases, the correct review is retrieved based on local trust and is blended into the results. This is why the results

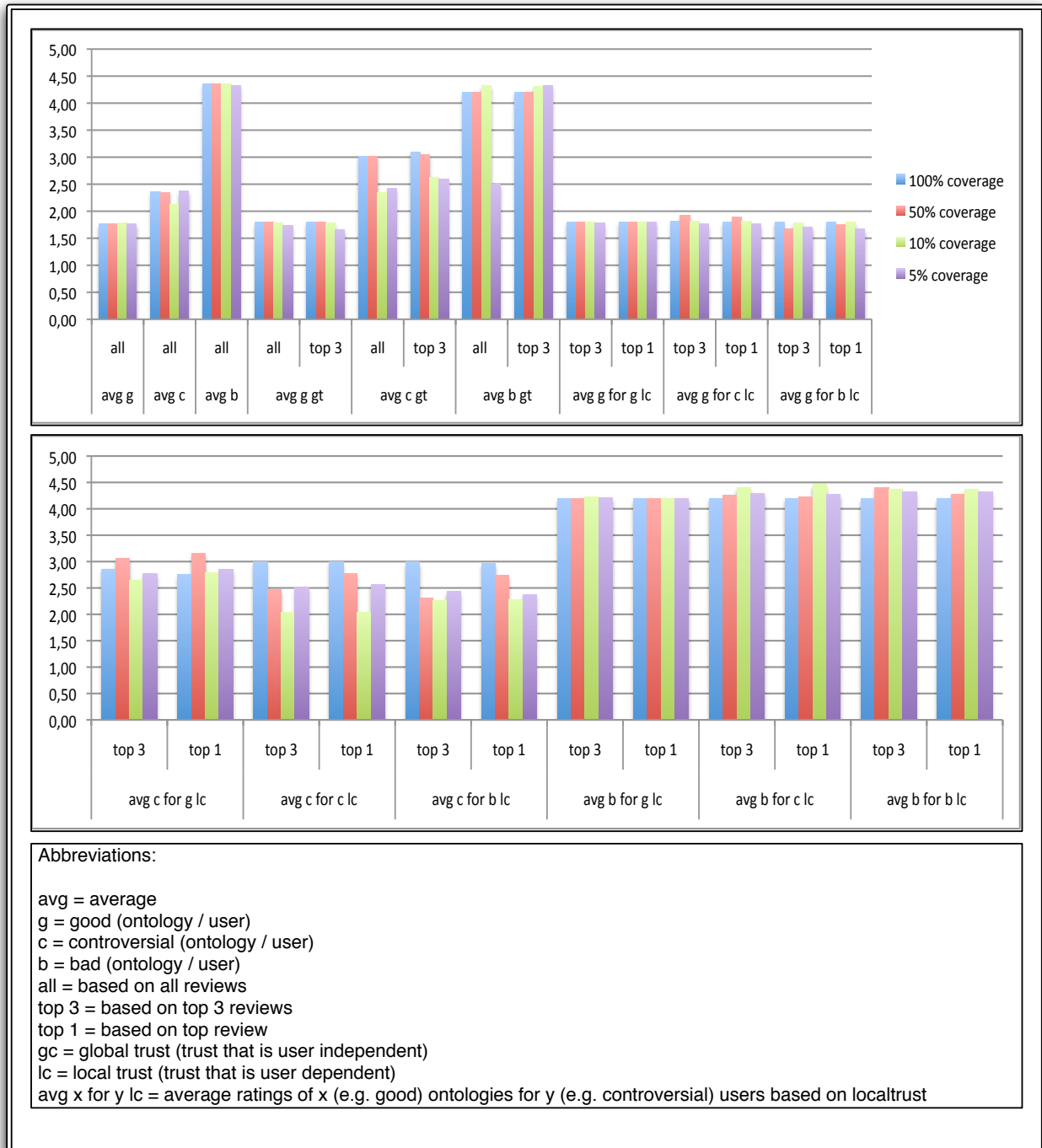


Figure 3.5: Second Scenario with 10% Error

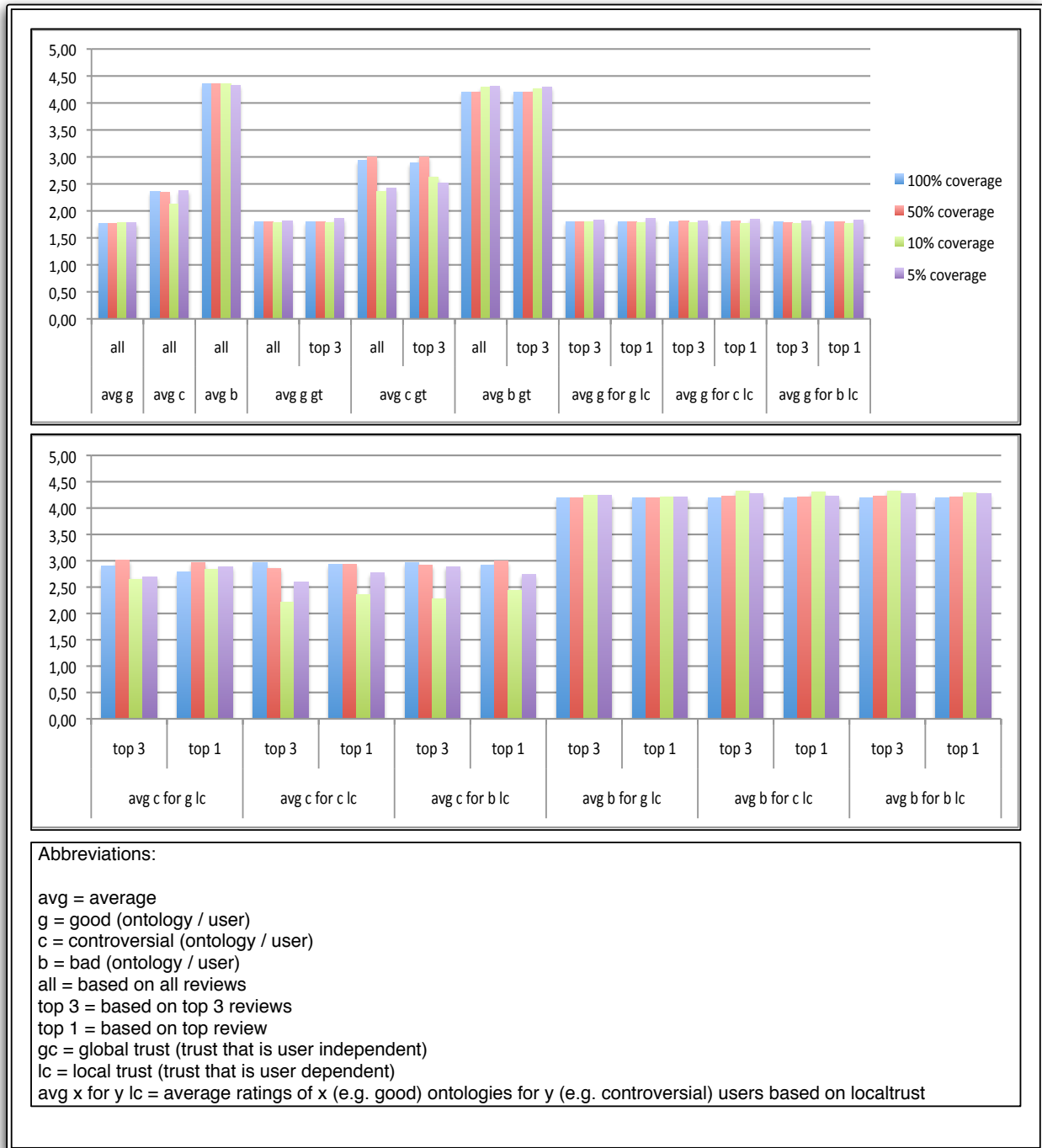


Figure 3.6: Second Scenario with 20% Error

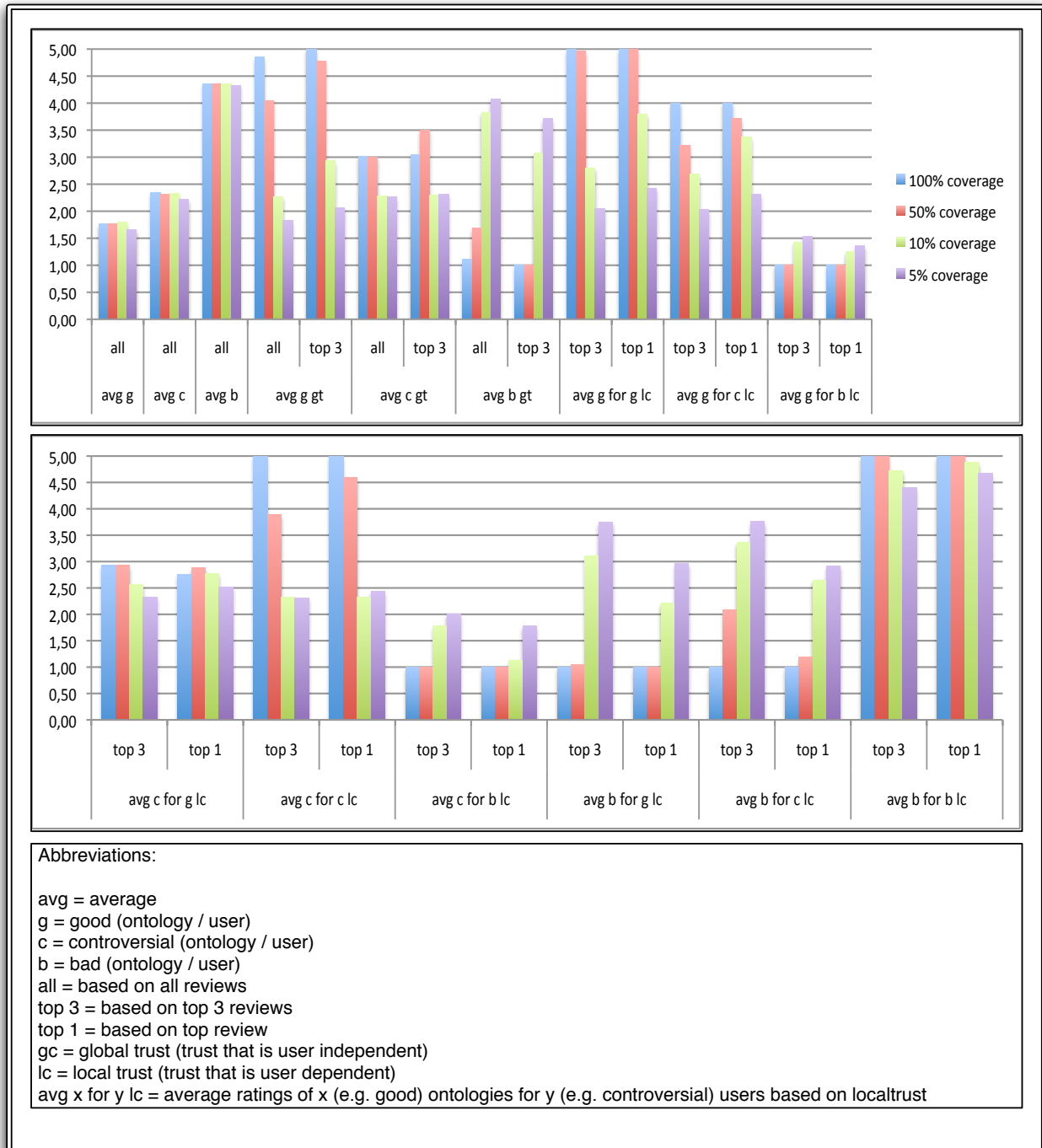


Figure 3.7: Third Scenario with 0% Error

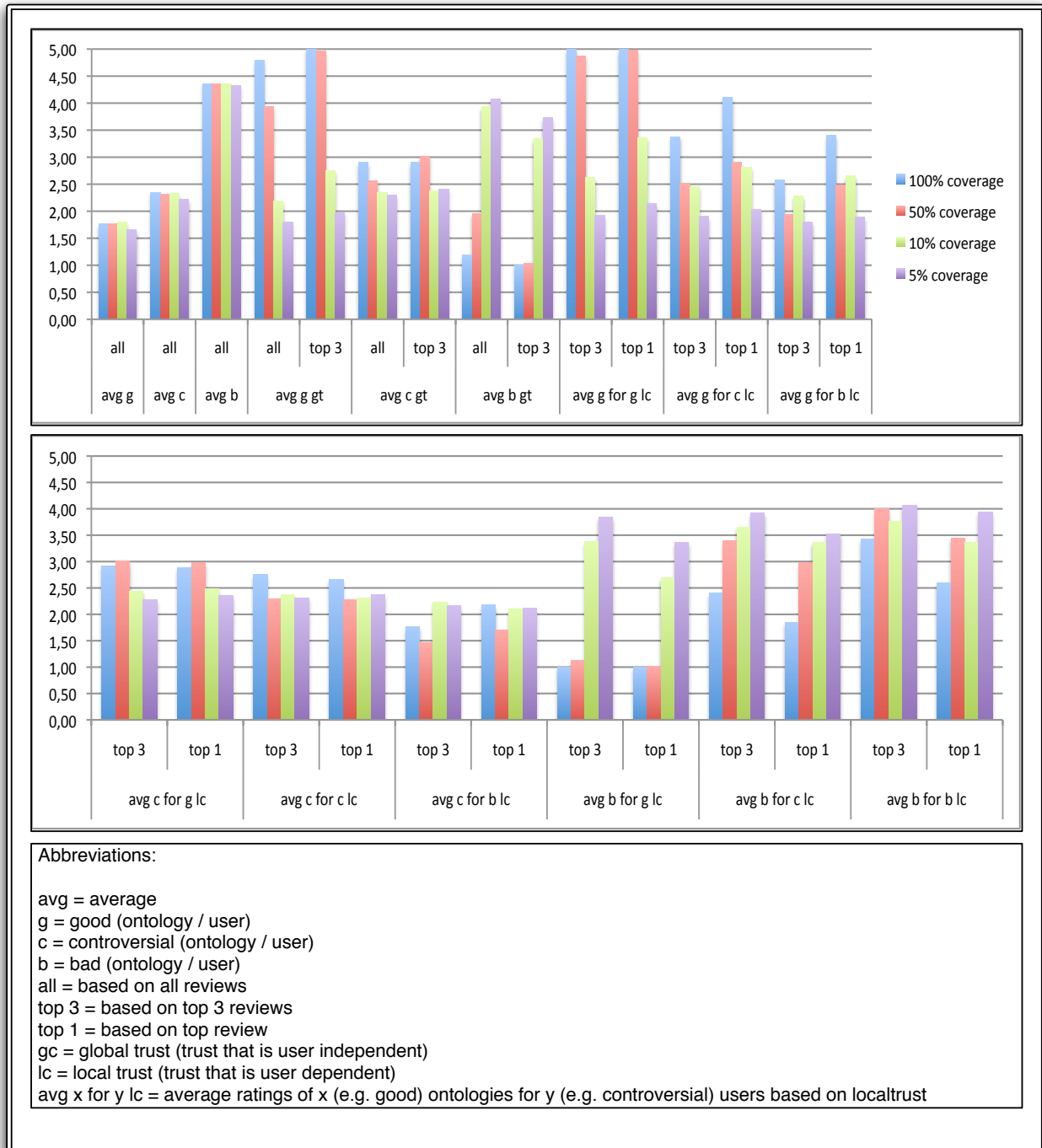


Figure 3.8: Third Scenario with 10% Error

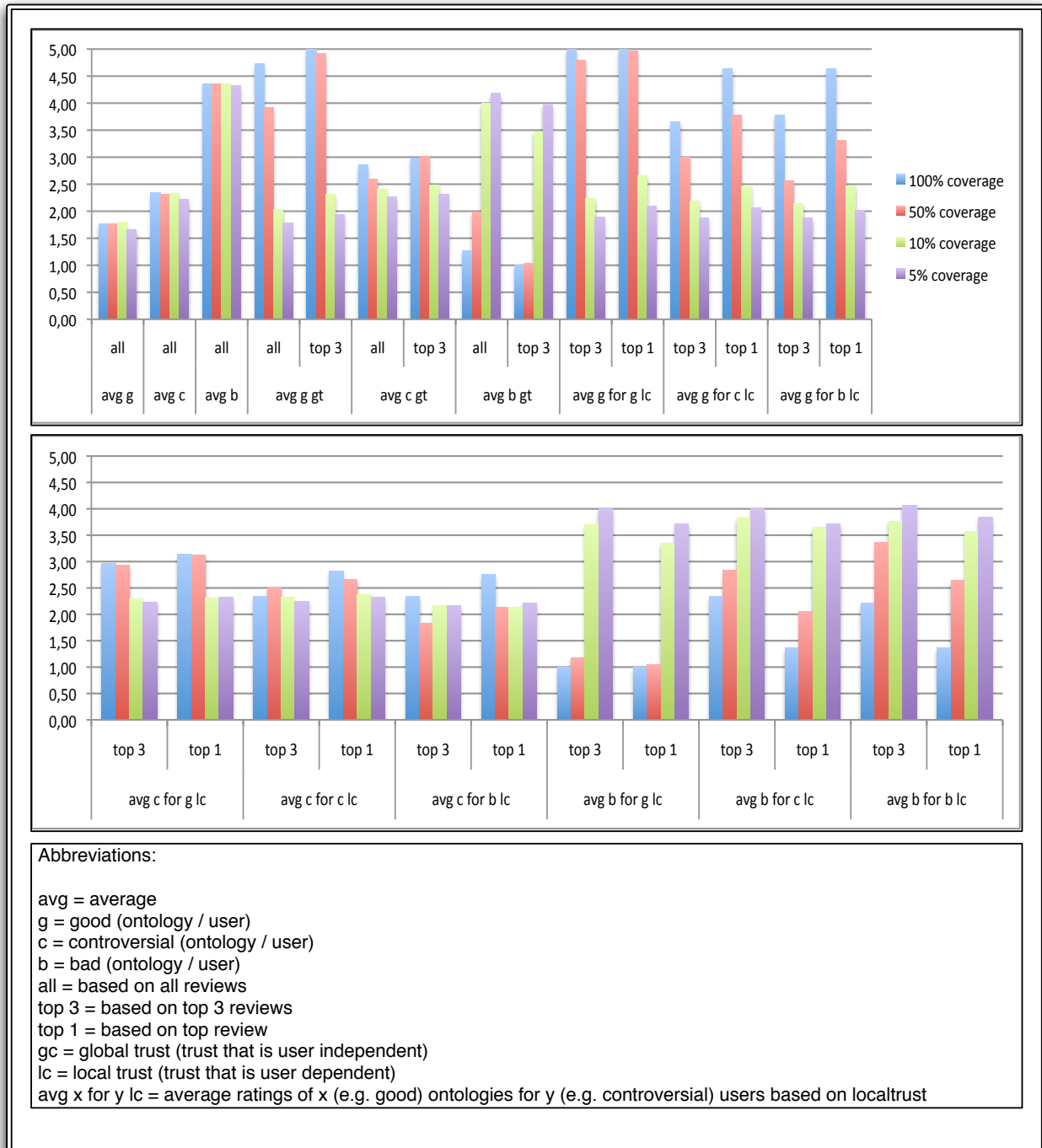


Figure 3.9: Third Scenario with 20% Error

deviate in direction of the expected results. Given a setting of 1000 users, 5% coverage and 0% error, there are almost no users disconnected from the WOT, as can be seen in Figure 3.12. If we remember that the minimal setting for achieving expected results is only partially dependent on the user size (the number of reviews required is independent from the number of users, but is based on the number of ontologies), we are not surprised to see improved results in the setting with more users (since the 5% here mean 10 times more reviews and trust statements than in the 100 user setting). As a conclusion of the comparison we note that the number of reviews and trust statement in total is not as important as having the right ones (the ones described in section 3.3.6). Given our experiment setup, by having a larger number of reviews and trust statements, we have achieved a better coverage and therefore better results.

3.3.6 Minimal Setting with Expected Behavior

As discussed above, most of the problems with results deviating from expectations are due to users that are not connected to the WOT, which causes the system to fall back on global trust. So the more users are not connected to the WOT, the more the results resemble global trust results. A minimum setting yielding perfect results (as expected or achieved with 100% coverage) for all users is achieved, when for each ontology at least one review exists from each user group (good, controversial, bad), and if each user trusts at least one review from a peer for each ontology in the system. This results in all users being connected to the WOT, and the reviewer from the peer group delivering the top-ranked result. Of course, if only one good review exists for each group, the results have to rely on the top review, not the top three reviews.

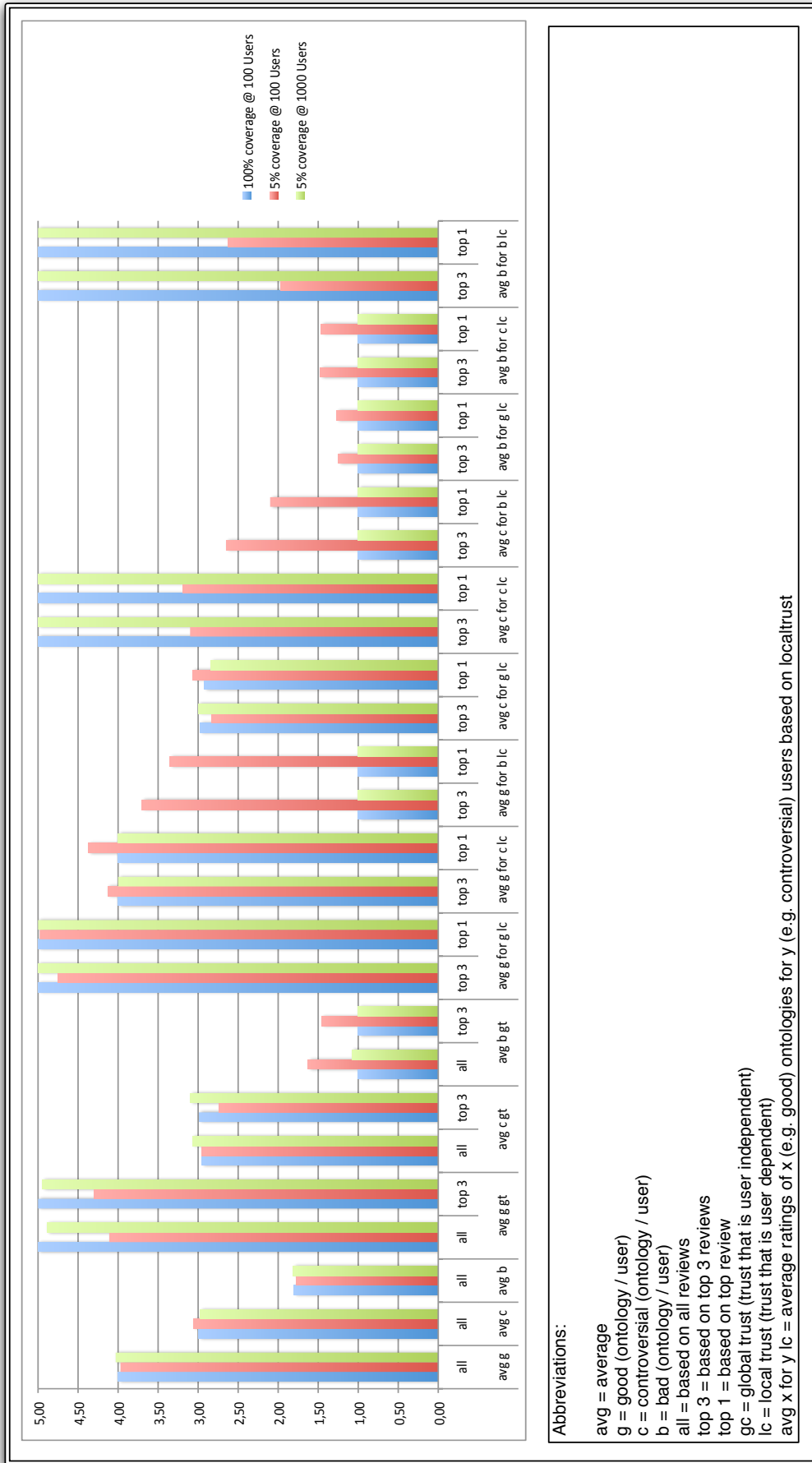


Figure 3.10: Comparison of Results Scenario 10% Error

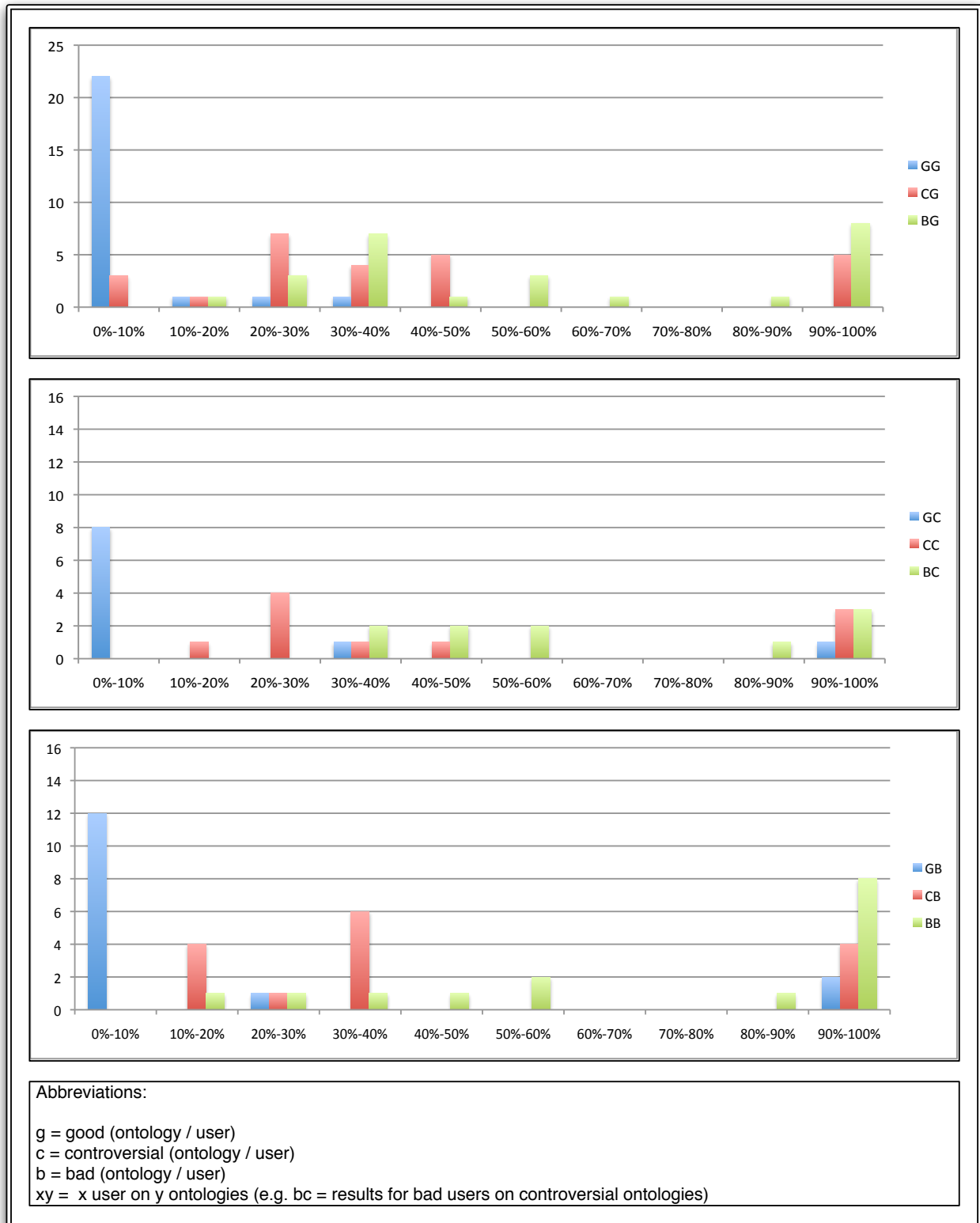


Figure 3.11: For this graph, we have checked for each ontology in the investigated group, which percentage of the users in the group investigated has no local trust information for this ontology. The findings were sorted into 10% buckets. The computations were performed with 100 users, 5% coverage and 0% error.

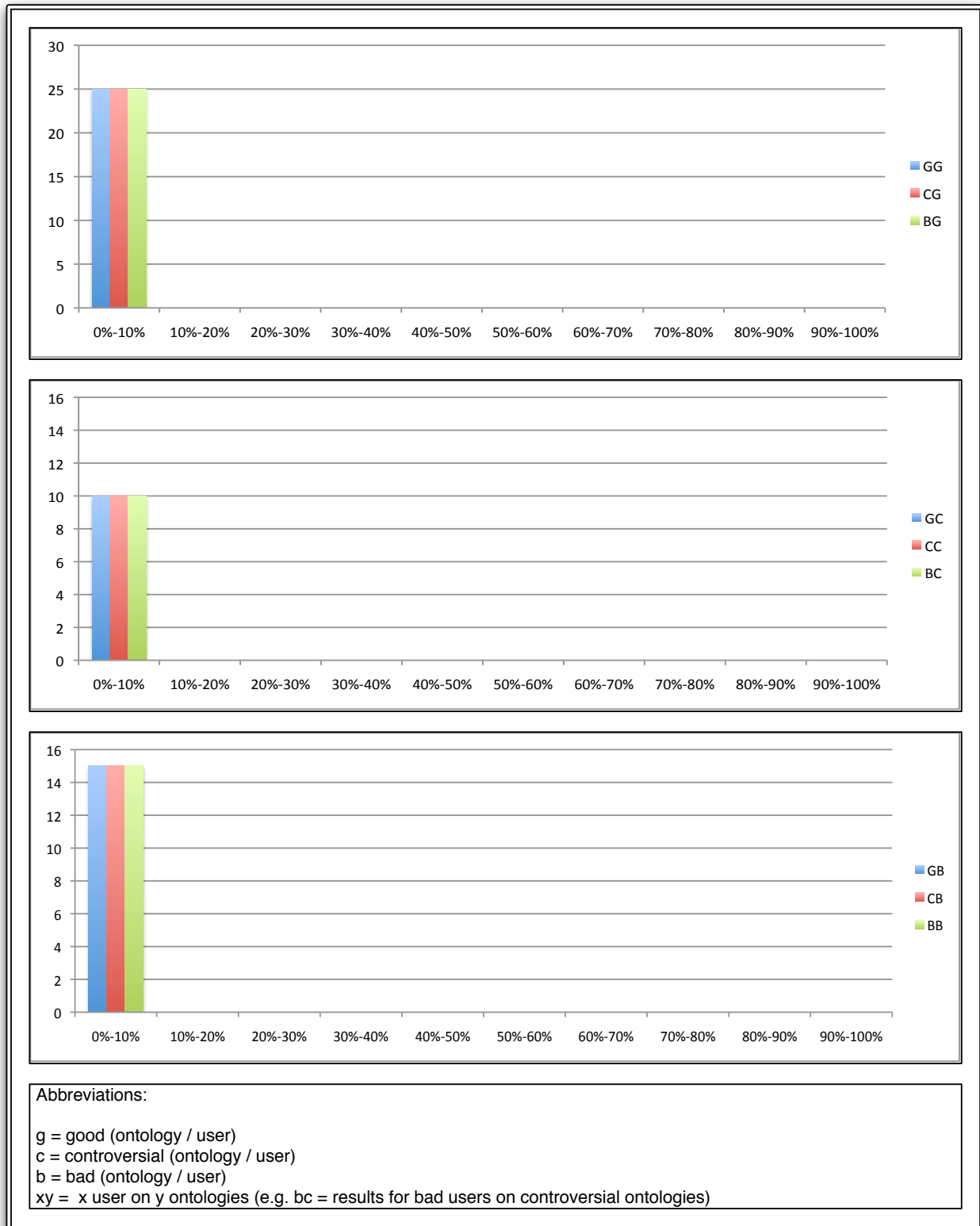


Figure 3.12: For this graph, we have checked for each ontology in the investigated group, which percentage of the users in the group investigated has no local trust information for this ontology. The findings were sorted into 10% buckets. The computations were performed with 1000 users, 5% coverage and 0% error.

3.4 Conclusion

In the concluding section we want to revise what we have learned from the simulations, and how this can help to improve the accuracy of ranking in real world systems like cupboard.

3.4.1 Validity of Simulation Results for Real World Systems

We think that the simulation we ran on the one hand shows that the algorithms do work in a perfect scenario, but also show how derivations can influence the results. We purposely chose to employ randomness when running the less dense scenarios and those that contained errors, because randomness is the worst case scenario. In real life, it is more likely that users will not trust or distrust random users, but users whose reviews they read. This would lead to a scenario which is more specific and will thus be closer to what we described as the minimal setting with expected behavior.

3.4.2 Accuracy of Ranking Results for the Individual

All our scenarios were based on computing the average over all ratings for all users of a certain group. This has influenced the results to our disadvantage, because in fact even for those cases the algorithms work as expected given the data available. They provided the expected results for the users connected to the WOT and thus having local trust available. For the rest, the system had to rely on global trust, which deviated from the expected local results. The more users in a setting were not connected to the WOT, the closer the results were to global trust. In a real world scenario, a perfect coverage would not be required, because not all users would want to know about all other ontologies. Most times, users have a clear idea what they need when searching for an ontology to reuse. In principle, a user will receive the correct score for each of the ontologies for which he has made a trust statement covering a review rating the ontology. So in the most ideal case, given that the user browsed the reviews for all properties of all ontologies and trusted one of the reviews per ontology–property connection, the rating results will be entirely based on local trust (i.e. the WOT), and will be as expected (see 3.3.1). Of course, this scenario is time-consuming and will likely not happen as such. So, the user must trust at least that many users on a metalevel (i.e. globally, per ontology or per property), that all ontology-property combinations are covered with reviews of trusted reviewers. If, for example, one reviewer reviews all the ontology-property connections the user is interested in, and the reviews can be trusted, a simple global trust statement from a user towards this one reviewer will be sufficient to provide the expected results. Especially in our data sparse setting, we can assume that in real world systems, the few existing trust statements would have been made by browsing ontologies of interest, and not randomly. Just by doing that, the local trust connectivity for the ontologies of interest would be higher than in our random scenario. We therefore argue that the results in real-world systems are better than those of our simulation using random behavior, simply because the users do not act randomly.

3.4.3 Significance of Error Settings

As mentioned before, when we tested for behavior of the ranking algorithms given erratic user behavior, we employed a random selection of users to trust and then had the users act in a certain percentage of the cases in contradiction to the rules defined (i.e. distrust instead or trust and vice versa). That means that a bad user would be trusted by a good user, but also that a good user might distrust another good user. While it was interesting to see how the results would be affected, we also argue that in reality, user behavior will not be that random and erratic. For any user not trying to act maliciously on purpose, there is no incentive to act in contrast to the best knowledge and, technically speaking, when a user enters a wrong statement into the system, the results of that behavior cannot be considered an error by the system since the user is responsible for providing correct data to the system.

3.4.4 Attacks on the System

As could be shown in the bad user scenario, the local trust algorithms are resistant to most kinds of attacks, which we simulated with the bad uses group. As long as the malicious subgroup stays separate from the rest of the users in the WOT, no harm can be done. In case the bad users are only a small group in relation to the overall group, the damage done is low even in the cases where they are accidentally trusted. But even introducing large scales of bad users into the system (making it the predominant group) does not influence the local trust settings for the user connected to the WOT. The global trust can be affected (compare to the link farms [WD05] on the web trying to increase their Google page rank), but it is only used for users who cannot be identified or are not connected to the WOT. Now one can argue that the bad users could try to gain the trust of normal users by acting like a good user in e.g. 90% of the cases, and then falsely review ontologies in the rest of the cases. This attack is not that easy, since we compute the WOT separately for each of the ontology-property combinations. That means that the bad user under normal circumstances would only be trusted in the cases where he is acting like a good user, and not in the other cases. The only way to draw an advantage would be by convincing a user to express a meta trust-statement, also covering potentially malicious reviews. But even if that is possible, one could argue that the good the bad user has done in order to gain trust is overall better for the system than the bad caused by having a few malicious cases. In order to prevent being a victim of such a malicious reviewer, users should be very sensitive as to who they trust on a meta-level.

3.4.5 Lessons Learned

It is important that users make use of the trust function in order to receive personalized results. If they do not, they receive ratings based on global trust, which works fine as long as most users are good users and the user has needs similar to the mainstream. It is furthermore important that users know only to issue a meta-trust statement when they are certain the user is deserving it. It should not be their default action to meta-trust a user. Because reviewers can have different strengths and weaknesses, it is important to use a system allowing to assign trust fine-grained, like the TS-ORS (see scenario 3 vs. scenario 2). Even though the local trust algorithms are immune to attacks from confined subgroups, it is in the best interest of the operator of a real-world TS-ORS to shut out bad users and keep the system clean.

Chapter 4

Method for Re-engineering Non-Ontological Resources into Ontologies

In this chapter we present an updated version of our method for re-engineering non-ontological resources into ontologies. The chapter supersedes the previous version of the method for re-engineering non-ontological resources presented in D2.2.2. We have opted for a pattern-based approach to carrying out the non-ontological re-engineering process. This method is applied to classification schemes, and thesauri.

With respect to our model for re-engineering non-ontological resources, in this deliverable we provide detailed descriptions of the:

- Software engineering abstraction levels.
- Ontology engineering abstraction levels.
- Transformation from the existing non-ontological resource to the target ontology.

With respect to the non-ontological resources re-engineering process, the changes over the last version of the process are mainly:

- The restructuring of the tasks for the non-ontological resource reverse engineering activity.
- The inclusion of a manual refinement task for the non-ontological resource transformation activity.

Regarding the patterns for re-engineering non-ontological resources, the changes over the last version of the patterns are mainly:

- The inclusion of patterns for the ABox transformation.
- The provision of formal transformations by using the formal definitions of the resources.
- The addition of external resources, e.g. Scarlet¹ and WordNet², to disambiguate the semantic relationships among the non-ontological resource elements.
- The use of logical patterns for the creation of *partOf*, and *subClassOf* relationships.
- The PR-NOR software library which implements the transformation process suggested by the patterns. This software library is described in deliverable *D2.5.2 Library of ontology design patterns, and software support for pattern-based design*.

¹<http://scarlet.open.ac.uk/>

²<http://wordnet.princeton.edu/>

4.1 Model for Re-engineering Non-Ontological Resources

In this section we describe our refined and improved model for re-engineering non-ontological resources. The previous version of our model was presented in [VTAGS⁺08]. The model for non-ontological resource re-engineering, depicted in Figure 4.1, is based on the software re-engineering model presented in [Byr92]. According to [Byr92] software re-engineering is the examination and alteration of a software system to re-constitute it in a new form and the subsequent implementation of the new form. The Figure also shows the following activities: NOR reverse engineering, NOR transformation, and ontology forward engineering.

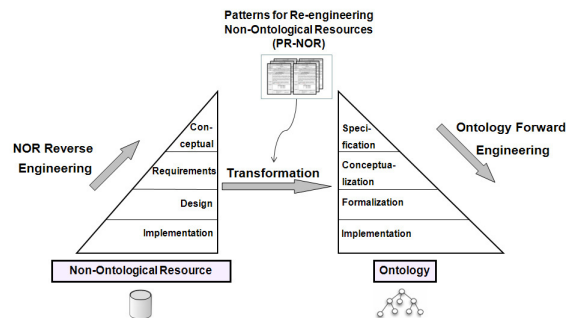


Figure 4.1: Re-engineering Model for Non-Ontological Resources

Since a software system consists of one or more programs, data files, databases, and job control scripts, we consider non-ontological resources as software resources, and therefore, we use the software abstraction levels shown in Figure 4.1 to depict the reverse engineering of the non-ontological resource. Understanding how non-ontological resource is created is useful for understanding how non-ontological resource can be re-engineered. The idea of levels of abstraction that underlies the development process also underlies the re-engineering process. This idea is used to model software development as a sequence of phases, where each phase corresponds to a particular level of abstraction.

In the left triangle of Figure 4.1 we can distinguish the four different abstraction levels to define each activity in software development:

1. *the conceptual abstraction level*, which describes in general terms the functional characteristics of the system;
2. *the requirements level*, which is the specification of the problem being solved;
3. *the design level*, which is the specification of the solution;
4. *the implementation level*, which refers to the coding, testing and delivery of the operational system.

As the level of abstraction decreases a system description becomes more detailed and the amount of information increases. Moreover, the higher the abstraction level the less information about a system there is to comprehend.

In the right triangle of Figure 4.1 we can distinguish the four different abstraction levels to define each activity in ontology engineering:

1. *the specification level*, which describes the collection of requirements that the ontology should fulfil;
2. *the conceptualization level*, which is the organization of the information from the acquisition process into meaningful conceptual models;
3. *the formalization level*, which represents the transformation of the conceptual model into a formal or semi-computable model according to a knowledge representation paradigm;

4. *the implementation level*, which refers to the generation of computable models according to the syntax of a formal representation language.

The model in Figure 4.1 suggests the path from the existing non-ontological resource to the target ontology. This transformation is guided by a set of Patterns for Re-engineering Non-Ontological Resources (PR-NOR), and goes from the non-ontological resource requirements/design level to the conceptualization level of the ontology.

4.2 Non-ontological Resources Re-engineering Process

The non-ontological resource re-engineering process consists of the activities depicted in Figure 4.2. With respect to the previous version of the process, presented in D2.2.2 [VTAGS⁺08], we restructured the tasks for the non-ontological resource reverse engineering activity and included a manual refinement task within the non-ontological resource transformation activity.

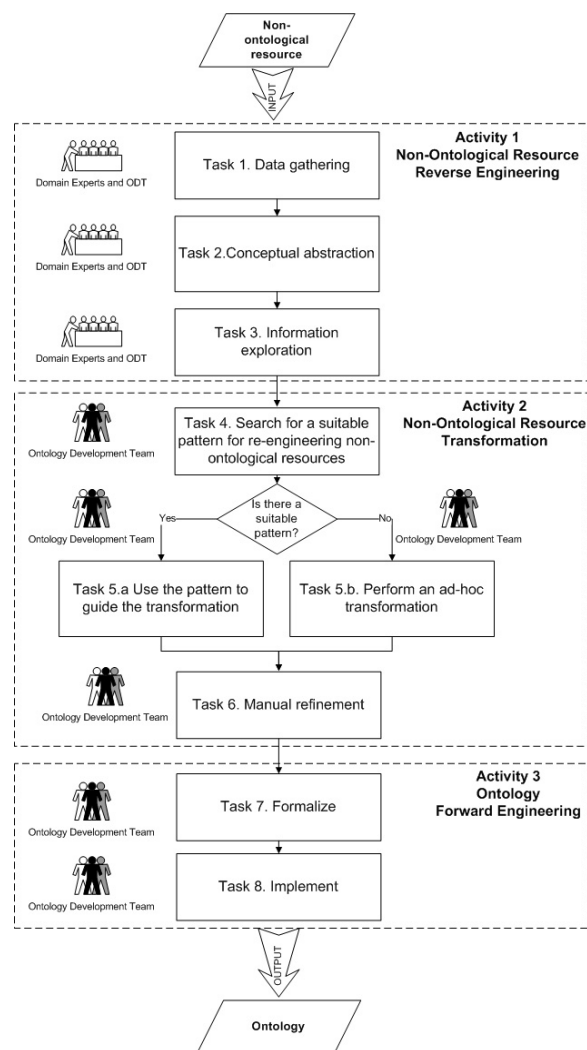


Figure 4.2: Re-engineering process for Non-Ontological Resources

1. **Non-Ontological Resource Reverse Engineering**, whose goal is to analyze a non-ontological resource to identify its underlying components and create representations of the resource at the different levels of abstraction (design, requirements and conceptual).

- **Task 1. Data gathering.** The goal of this task is to search and compile all the available data and documentation about the non-ontological resource including purpose, components, data model and implementation details.
 - **Task 2. Conceptual abstraction.** The goal of this task is to identify the schema of the non-ontological resource including the conceptual components and their relationships. If the conceptual schema is not available in the documentation, the schema should be reconstructed manually or by using a data modelling tool.
 - **Task 3. Information exploration.** The goal of this task is to find out how the conceptual schema of the non-ontological resource and its content are represented in the data model. If the non-ontological resource data model is not available in the documentation, the data model should be reconstructed manually or by using a data modelling tool.
2. **Non-Ontological Resource Transformation**, whose goal is to generate a conceptual model from the non-ontological resource. We propose the use of Patterns for Re-engineering Non-Ontological Resources (PR-NOR) to guide the transformation process.
- **Task 4. Search for a suitable pattern for re-engineering non-ontological resource.** The goal of this task is to find out if there is any applicable re-engineering pattern to transform the non-ontological resource into a conceptual model. To search for a suitable pattern for re-engineering non-ontological resource the NeOn library of patterns³ can be used. First, the non-ontological resource type has to be identified. Second, the internal data model of the non-ontological resource has to be identified as well. Third, the transformation approach has to be selected, from the ones described in section 4.3.1.
 - **Task 5.a. Use patterns for re-engineering to guide the transformation.** The goal of this task is to apply the re-engineering pattern obtained in task 4 to transform the non-ontological resource into a conceptual model. If a suitable pattern for re-engineering non-ontological resource is found then the conceptual model is created from the non-ontological resource following the procedure established in the pattern for re-engineering.
 - **Task 5.b. Perform and ad-hoc transformation.** The goal of this task is to set up an *ad-hoc* procedure to transform the non-ontological resource into a conceptual model, when a suitable pattern for re-engineering was not found. This *ad-hoc* procedure may be generalized to create a new pattern for re-engineering non-ontological resource.
 - **Task 6. Manual refinement.** The goal of this task is to check if some inconsistency is present after the transformation. Software developers and ontology practitioners with the support of domain experts can fix manually some generated inconsistencies after the transformation.
3. **Ontology Forward Engineering**, whose goal is to generate the ontology. We use the ontology levels of abstraction to depict this activity because they are directly related to the ontology development process.
- **Task 7. Formalize.** The goal of this task is to transform the conceptual model obtained in task 5.a or 5.b into a formalized model, according to a knowledge representation paradigm as description logics, first order logic, etc.
 - **Task 8. Implement.** The goal of this task is the ontology implementation in an ontology language.

4.3 Patterns for Re-engineering NORs into Ontologies

Patterns for re-engineering non-ontological resources (PR-NOR) define a procedure that transforms the non-ontological resource components into ontology representational primitives. To this end, patterns take advan-

³<http://www.ontologydesignpatterns.org>

tage of the non-ontological resource underlying data model. The data model defines how the different components of the non-ontological resource are represented. According to the non-ontological resource categorization presented in [VTAGS⁺08], the data model can be different even for the same type of non-ontological resource. For every data model we can define a process with a well-defined sequence of activities to extract the non-ontological resources components, and then to map these components to a conceptual model of an ontology. Each of these processes can be expressed as a pattern for re-engineering non-ontological resources.

Next, we describe the changes over the previous version of the patterns, i.e. the inclusion of patterns for the ABox transformation, the provision of formal transformations by using the formal definitions of the resources, the addition of external resources to disambiguate the semantic relationships among the non-ontological resource elements, and the use of logical patterns for the creation of *partOf*, and *subClassOf* relationships.

4.3.1 Transformation approaches

In the previous version of this deliverable, D2.2.2 [VTAGS⁺08], we have identified three transformation approaches for re-engineering non-ontological resources into ontologies, these approaches are shown in Figure 4.3 and are the following:

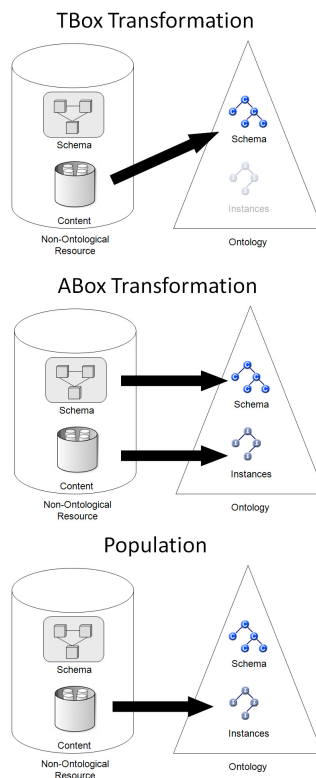


Figure 4.3: Transformation approaches

- *TBox transformation*: for transforming the resource content into an ontology schema. This transformation approach tries to enforce a formal semantics to the re-engineered resources, even at the cost of changing their structure [SAd⁺07a].
- *ABox transformation*: for transforming the resource schema into an ontology schema, and the resource content, into ontology instances. This transformation approach leaves the informal semantics of the re-engineered resources mostly untouched [SAd⁺07a].
- *Population*: for transforming the resource content into instances of an existing ontology.

In D2.2.2 [VTAGS⁺08], we have mainly presented patterns for the TBox transformation approach. In this deliverable we include patterns for the ABox transformation approach.

4.3.2 Formalization of the resources

In this section we provide a formal definition of the resources the patterns are dealing with. These resources are ontologies, classification schemes and thesauri.

Ontology

Based on the definition provided in [ES07], we can define a lightweight ontology O as the following tuple:

$$O = \langle OS, KB \rangle$$

Where OS represents the ontology schema, and KB represents the knowledge base.

An ontology schema OS is defined through the following tuple:

$$OS = \langle C, A, R, S \rangle$$

where:

- $C = \{c_1, \dots, c_n\}$, a finite set of concepts.
- $A = \{a_1, \dots, a_n\}$, a finite set of attributes, where every $a_i \subseteq C \times \text{Literal}$.
- $R = \{r_1, \dots, r_n\}$, a finite set of binary relations, where every $r_i \subseteq C \times C$.
- $S : C \rightarrow C$, a *subClassOf* relation.

A knowledge base is a structure:

$$KB = \langle C, A, R, I, t_C, t_A, t_R \rangle$$

consisting of:

- three sets C , A , and R as defined before.
- a set $I = \{i_1, \dots, i_n\}$ whose elements are called instance identifiers
- a function $t_C : C \rightarrow I$ called concept instantiation
- a function $t_A : A \rightarrow I$ called attribute instantiation
- a function $t_R : R \rightarrow I^2$ called relation instantiation

Classification Scheme

A classification scheme [ISO04] is the descriptive information for an arrangement or division of objects into groups based on characteristics, which the objects have in common. We formally define a classification scheme as the following tuple:

$$CS = \langle CSS, CSD \rangle$$

Where CSS represents the schema of the classification scheme, and CSD represents the content of the classification scheme.

The schema of the classification scheme, CSS , is defined as:

$$CSS = \langle CSC, CSA, CSR \rangle$$

where:

- $CSC = \{c_1\}$, a set of one category.
- $CSA = \{a_1, \dots, a_n\}$, a finite set of attributes, where every $a_i \subseteq CSC \times Literal$.
- $CSR = \{r_1\}$, a set of one binary relation, where $r_1 \subseteq CSC \times CSC$.

The content of the classification scheme, CSD , is defined as:

$$CSD = \langle CSC, CSA, CSR, CSI, CSt_C, CSt_A, CSt_R \rangle$$

consisting of:

- three sets CSC , CSA and CSR as defined before.
- a set $CSI = \{csi_1, \dots, csi_n\}$ whose elements are called classification scheme item identifiers
- a function $CSt_C : CSC \rightarrow CSI$ called classification scheme item instantiation
- a function $CSt_A : CSA \rightarrow CSI$ called classification scheme attribute instantiation
- a function $CSt_R : CSR \rightarrow CSI^2$ called classification scheme relation instantiation

Thesaurus

According to the ANSI/NISO Z39.19-2005 [ANS05], a thesaurus is a controlled vocabulary in a known order and structured so that various relationships among terms are displayed clearly and identified by standardized relationships indicators. We formally define a thesaurus as the following tuple:

$$T = \langle TS, TD \rangle$$

Where TS represents the schema of the thesaurus, and TD represents the content of the thesaurus.

The schema of the thesaurus is defined as:

$$TS = \langle TST, TSA, TSB, TSN, TSR \rangle$$

where:

- $TST = \{tst_{pt}, tst_{npt}\}$, a set of two categories, tst_{pt} preferred term, and tst_{npt} , non-preferred term.
- $TSA = \{a_1, \dots, a_n\}$, a finite set of attributes, where every $a_i \subseteq TST \times Literal$.
- $TSB = \{tsb_1, \dots, tsb_n\}$, a finite set of *broader term* relations.
- $TSN = \{tsn_1, \dots, tsn_n\}$, a finite set of *narrower term* relations.
- $TSR = \{tsr_1, \dots, tsr_n\}$, a finite set of *related term* relations.

The content of the thesaurus is defined as:

$$TD = \langle TST, TSA, TSB, TSN, TSR, CSt_C, CSt_A, CSt_R \rangle$$

consisting of:

- five sets TST , TSA , TSB , TSN , and TSR as defined before.
- a set $TI = \{ti_1, \dots, ti_n\}$ whose elements are called thesaurus term identifiers
- a function $Tt_t : TST \rightarrow TI$ called thesaurus term instantiation
- a function $Tt_a : TSA \rightarrow TI$ called thesaurus attribute instantiation
- a function $Tt_b : TSB \rightarrow TI^2$ called thesaurus *broader term* relation instantiation
- a function $Tt_n : TSN \rightarrow TI^2$ called thesaurus *narrower term* relation instantiation
- a function $Tt_r : TSR \rightarrow TI^2$ called thesaurus *related term* relation instantiation

4.3.3 Semantics of the relations among the non-ontological resource entities

Patterns for the TBox transformation approach must disambiguate the semantics of the relations among the non-ontological resource entities. To perform this disambiguation, patterns rely on external resources. At this stage they are using:

- Scarlet⁴, which discovers semantic relations between concepts by using the entire Semantic Web as a source of background knowledge: it automatically identifies and explores multiple and heterogeneous online ontologies to derive relations.
- WordNet⁵, which is a lexical database for the English language. Patterns use the WordNet relations for discovering the semantic relations between concepts, when possible.

In cases where the external resource gives an empty result, i.e. it does not provide any relation between two concepts, the patterns take advantage of the use of logical patterns⁶ for asserting the relation *partOf* or *subClassOf* as appropriate.

4.3.4 Template for the PR-NOR

In this section we present the proposed template used to describe the patterns for re-engineering non-ontological resources (PR-NOR). To present the patterns for re-engineering non-ontological resources we have modified the tabular template used in [VTAGS⁺08]. The template adapted and the meaning of each field is shown in Table 4.1.

Table 4.1: Pattern for Re-engineering Non-Ontological Resource Template

Slot	Value
General Information	
Name	Name of the pattern
Identifier	An acronym composed of component type + abbreviated name of the component + number
Component Type	Pattern for Re-engineering Non-Ontological Resource (PR-NOR)
Use Case	
General	Description in natural language of the re-engineering problem addressed by the pattern for re-engineering non-ontological resources.
Example	Description in natural language of an example of the re-engineering problem.
Pattern for Re-engineering Non-Ontological Resource	

⁴Relation Discovery on The Semantic Web <http://scarlet.open.ac.uk>

⁵<http://wordnet.princeton.edu/>

⁶Logical patterns are included in the ODP portal at <http://ontologydesignpatterns.org>

Table 4.1: Pattern for Re-engineering Non-Ontological Resource Template (Continued)

Slot	Value
INPUT: Resource to be Re-engineered	
General	Description in natural language of the non-ontological resource.
Example	Description in natural language of an example of the non-ontological resource.
Graphical Representation	
General	Graphical representation of the non-ontological resource.
Example	Graphical representation of the example of non-ontological resource.
OUTPUT: Designed Ontology	
General	Description in natural language of the ontology created after applying the pattern for re-engineering the non-ontological resource.
Graphical Representation	
(UML) General Solution Ontology	Graphical representation, using the UML profile [BH06], of the ontology created for the non-ontological resource being re-engineered.
(UML) Example Solution Ontology	Example showing a graphical representation, using the UML profile [BH06], of the ontology created for the non-ontological resource being used.
PROCESS: How to Re-engineer	
General	Description in natural language of the general re-engineering process, using a sequence of activities.
Example	Description in natural language of the re-engineering process applied to the non-ontological resource example, using the above sequence of activities.
Formal Transformation	
General	Formal description of the transformation by using the formal definitions of the resources.
Relationships (Optional)	
Relations to other modelling components	Description of any relation to other PR-NOR patterns or other ontology design patterns.

4.4 PR-NOR Library

In this section we present the re-engineering patterns (PR-NOR) for re-engineering classification schemes and thesauri into ontologies. We include the patterns for the TBox and ABox transformations.

4.4.1 Patterns for Re-engineering Classification Schemes into Ontologies

In this section we present the re-engineering patterns (PR-NOR) for re-engineering classification schemes into ontologies. The patterns are:

- Patterns for the TBox transformations
 - PR-NOR-CLTX-01. Pattern for re-engineering a classification scheme, which follows the path enumeration data model, into an ontology schema.
 - PR-NOR-CLTX-02. Pattern for re-engineering a classification scheme, which follows the adjacency list data model, into an ontology schema.
 - PR-NOR-CLTX-03. Pattern for re-engineering a classification scheme, which follows the snowflake data model, into an ontology schema.
 - PR-NOR-CLTX-04. Pattern for re-engineering a classification scheme, which follows the flattened data model, into an ontology schema.
- Patterns for the ABox transformations

- PR-NOR-CLLO-10. Pattern for re-engineering a classification scheme, which follows the path enumeration data model, into an ontology.
- PR-NOR-CLLO-11. Pattern for re-engineering a classification scheme, which follows the adjacency list data model, into an ontology.
- PR-NOR-CLLO-12. Pattern for re-engineering a classification scheme, which follows the snowflake data model, into an ontology.
- PR-NOR-CLLO-13. Pattern for re-engineering a classification scheme, which follows the flattened data model, into an ontology.

Patterns for the TBox transformations

These patterns transform the resource content into an ontology schema. The TBox transformation approach tries to enforce a formal semantics to the re-engineered resources, even at the cost of changing their structure [SAd⁺07a]. For the disambiguation of the semantics of the relations among classification scheme items, the patterns rely on an external resource, e.g. Scarlet and WordNet. In the case of that the external resource does not provide any relation between two items, the patterns take advantage of the use of logical patterns⁷ asserting the relation *partOf* or *subClassOf*.

PR-NOR-CLTX-01. Pattern for re-engineering a classification scheme, which follows the path enumeration data model, into an ontology schema. The pattern for re-engineering non-ontological resource shown in Table 4.2 suggests a guide to transform a classification scheme into an ontology schema. The classification scheme is modeled/represented with a path enumeration data model.

Table 4.2: Pattern for re-engineering a classification scheme, which follows the path enumeration data model, into an ontology schema.

Slot	Value
General Information	
Name	Pattern for Re-engineering a Classification Scheme, which follows the Path Enumeration Data Model, into an Ontology Schema.
Identifier	PR-NOR-CLTX-01
Type of Component	Pattern for Re-engineering Non-Ontological Resource (PR-NOR)
Use Case	
General	Re-engineering a classification scheme, which follows the path enumeration model, to design an ontology schema.
Example	Suppose that someone wants to build an ontology based on the International Standard Classification of Occupations (for European Union purposes) ISCO-88 (COM). This classification scheme follows the path enumeration data model.
Pattern for Re-engineering Non-Ontological Resource	
INPUT: Resource to be Re-engineered	
General	A non-ontological resource holds a classification scheme which follows the path enumeration model. A classification scheme is a rooted tree of concepts, in which each concept groups entities by some particular degree of similarity. The semantics of the hierarchical relation between parents and children concepts may vary depending of the context. The path enumeration data model [Bra05], for classification schemes, takes advantage of that there is one and only one path from the root to every item in the classification. The path enumeration model stores that path as string by concatenating either the edges or the keys of the classification scheme items in the path.
Example	The International Standard Classification of Occupations (for European Union purposes), 1988 version: ISCO-88 (COM) published by Eurostat is modeled with the path enumeration data model. This classification scheme is available at http://ec.europa.eu/eurostat/ramon/
Graphical Representation	

⁷Logical patterns are included in the ODP portal at <http://ontologydesignpatterns.org>

Table 4.2: Pattern for re-engineering a classification scheme, which follows the path enumeration data model, into an ontology schema.(continued)

Slot	Value																					
General	<table border="1"> <thead> <tr> <th style="background-color: black; color: white;">Path Enumeration</th> <th style="background-color: black; color: white;">CS Name</th> <th style="background-color: black; color: white;">CS Description</th> </tr> </thead> <tbody> <tr><td>1</td><td>Name1</td><td>Description1</td></tr> <tr><td>11</td><td>Name11</td><td>Description11</td></tr> <tr><td>111</td><td>Name111</td><td>Description111</td></tr> <tr><td>12</td><td>Name12</td><td>Description12</td></tr> <tr><td>2</td><td>Name2</td><td>Description2</td></tr> <tr><td>...</td><td>...</td><td>...</td></tr> </tbody> </table>	Path Enumeration	CS Name	CS Description	1	Name1	Description1	11	Name11	Description11	111	Name111	Description111	12	Name12	Description12	2	Name2	Description2
Path Enumeration	CS Name	CS Description																				
1	Name1	Description1																				
11	Name11	Description11																				
111	Name111	Description111																				
12	Name12	Description12																				
2	Name2	Description2																				
...																				
Example	<table border="1"> <thead> <tr> <th style="background-color: black; color: white;">Code</th> <th style="background-color: black; color: white;">Level</th> <th style="background-color: black; color: white;">Name</th> </tr> </thead> <tbody> <tr><td>1</td><td>1</td><td>LEGISLATORS, SENIOR OFFICIALS AND MANAGERS</td></tr> <tr><td>11</td><td>2</td><td>Legislators and senior officials</td></tr> <tr><td>111</td><td>3</td><td>Corporate managers</td></tr> <tr><td>2</td><td>1</td><td>PROFESSIONALS</td></tr> <tr><td>...</td><td>...</td><td>...</td></tr> </tbody> </table>	Code	Level	Name	1	1	LEGISLATORS, SENIOR OFFICIALS AND MANAGERS	11	2	Legislators and senior officials	111	3	Corporate managers	2	1	PROFESSIONALS			
Code	Level	Name																				
1	1	LEGISLATORS, SENIOR OFFICIALS AND MANAGERS																				
11	2	Legislators and senior officials																				
111	3	Corporate managers																				
2	1	PROFESSIONALS																				
...																				
OUTPUT: Designed Ontology																						
General	<p>The generated ontology will be based on the taxonomy architectural pattern (AP-TX-01) [SFBG⁺07]. Each category in the classification scheme is mapped to a class, and the semantics of the relationship between children and parent categories are disambiguated by using an external resource. In the case of that the external resource does not provide any relation between two items, the pattern takes advantage of the use of logical patterns for asserting the relation <i>partOf</i> or <i>subClassOf</i>.</p>																					
Graphical Representation																						
(UML) General Solution Ontology	<pre> classDiagram Class0 < -- Class1 Class0 < -- Class2 Class1 < -- Class3 Class1 < -- Class4 Class3 < -- Class5 Class4 < -- Class6 </pre>																					
(UML) Example Solution Ontology	<pre> classDiagram Occupation < -- LEGISLATORS_SENIOR_OFFICIALS_AND_MANAGERS Occupation < -- PROFESSIONALS LEGISLATORS_SENIOR_OFFICIALS_AND_MANAGERS < -- Legislators_and_senior_officials LEGISLATORS_SENIOR_OFFICIALS_AND_MANAGERS < -- Corporate_managers </pre>																					
PROCESS: How to Re-engineer																						

Table 4.2: Pattern for re-engineering a classification scheme, which follows the path enumeration data model, into an ontology schema.(continued)

Slot	Value
General	<ol style="list-style-type: none"> 1. Identify the classification scheme items whose their path enumeration values have the shortest length, i.e. classification scheme items without parents. 2. For each one of the above identified classification scheme items ce_i: <ol style="list-style-type: none"> 2.1. Create the corresponding ontology class, C_i class. 2.2. Identify the classification scheme items, ce_j, which are children of ce_i, by using the path enumeration values. 2.3. For each one of the above identified classification scheme items ce_j: <ol style="list-style-type: none"> 2.3.1. Create the corresponding ontology class, C_j class. 2.3.2. Using the external resource identify the semantics of the relation between C_j and C_i and set up the relation identified. 2.3.3. Repeat from step 2.2 for ce_j as a new ce_i. 3. If there are more than one classification scheme items without parent ce_i <ol style="list-style-type: none"> 3.1. Create an <i>ad-hoc</i> class as the root class of the ontology. 3.2. Using the external resource identify the semantics of the relation between C_i class and the root class, and set up the relation identified.
Example	<ol style="list-style-type: none"> 1. Create the <code>LEGISLATORS, SENIOR OFFICIALS AND MANAGERS</code> class. <ol style="list-style-type: none"> 1.1. Create the <code>Legislators</code> and <code>senior officials</code> class. 1.2. Using the external resource identify the semantics of the relation between <code>Legislators</code> and <code>senior officials</code> and <code>LEGISLATORS, SENIOR OFFICIALS AND MANAGERS</code> and set up the relation identified. 1.3. Create the <code>Corporate managers</code> class. 1.4. Using the external resource identify the semantics of the relation between <code>Corporate managers</code> and <code>LEGISLATORS, SENIOR OFFICIALS AND MANAGERS</code> and set up the relation identified. 2. Create the <code>PROFESSIONALS</code> class. 3. Create the <code>Occupation</code> class. 4. Using the external resource identify the semantics of the relation between <code>LEGISLATORS, SENIOR OFFICIALS AND MANAGERS</code> and <code>Occupation</code> and set up the relation identified. 5. Using the external resource identify the semantics of the relation between <code>PROFESSIONALS</code> and <code>Occupation</code> and set up the relation identified.
Formal Transformation	
General	Classification Scheme: $CS = \langle CSS, CSD \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $CSD \rightarrow OS$
Relationships	
Relations to other modelling components	Use the Architectural Pattern: TX-AP-01 [SFBG ⁺ 07]

PR-NOR-CLTX-02. Pattern for re-engineering a classification scheme, which follows the adjacency list data model, into an ontology schema The pattern for re-engineering non-ontological resource shown in Table 4.3 suggests a guide to transform a classification scheme into an ontology schema. The classification scheme is modeled/represented with an adjacency list data model.

Table 4.3: Pattern for re-engineering a classification scheme, which follows the adjacency list data model, into an ontology schema

Slot	Value
General Information	

Table 4.3: Pattern for re-engineering a classification scheme, which follows the adjacency list data model, into an ontology schema(continued)

Slot	Value																														
Name	Pattern for Re-engineering a Classification Scheme, which follows the Adjacency List Data Model, into an Ontology Schema																														
Identifier	PR-NOR-CLTX-02																														
Type of Component	Pattern for Re-engineering Non-Ontological Resource (PR-NOR)																														
Use Case																															
General	Re-engineering a classification scheme, which follows the adjacency list model, to design an ontology schema.																														
Example	Suppose that someone wants to build an ontology based on the water areas classification published by FAO. This classification scheme follows the adjacency list data model.																														
Pattern for Re-engineering Non-Ontological Resource																															
INPUT: Resource to be Re-engineered																															
General	A non-ontological resource holds a classification scheme which follows the adjacency list model. A classification scheme is a rooted tree of concepts, in which each concept groups entities by some particular degree of similarity. The semantics of the hierarchical relation between parents and children concepts may vary depending of the context. The adjacency list data model [Bra05] for hierarchical classifications proposes to create an entity which holds a list of items with a linking column associated to their parent items.																														
Example	The FAO classification for water areas groups them according to some different criteria as environment, statistics, and jurisdiction, among others. This classification scheme is available at http://www.fao.org/figis/servlet/RefServlet																														
Graphical Representation																															
General	<table border="1"> <thead> <tr> <th>ID</th> <th>CS Name</th> <th>Parent ID</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Category1</td> <td>Null</td> </tr> <tr> <td>2</td> <td>Category2</td> <td>Null</td> </tr> <tr> <td>3</td> <td>Category3</td> <td>1</td> </tr> <tr> <td>4</td> <td>Category4</td> <td>1</td> </tr> <tr> <td>5</td> <td>Category6</td> <td>3</td> </tr> <tr> <td>6</td> <td>Category7</td> <td>4</td> </tr> <tr> <td>...</td> <td>...</td> <td>...</td> </tr> </tbody> </table>	ID	CS Name	Parent ID	1	Category1	Null	2	Category2	Null	3	Category3	1	4	Category4	1	5	Category6	3	6	Category7	4						
ID	CS Name	Parent ID																													
1	Category1	Null																													
2	Category2	Null																													
3	Category3	1																													
4	Category4	1																													
5	Category6	3																													
6	Category7	4																													
...																													
Example	<table border="1"> <thead> <tr> <th>ID</th> <th>CSI Name</th> <th>Parent ID</th> </tr> </thead> <tbody> <tr> <td>20000</td> <td>Water area</td> <td></td> </tr> <tr> <td>21000</td> <td>Environmental area</td> <td>20000</td> </tr> <tr> <td>24020</td> <td>Jurisdiction area</td> <td>20000</td> </tr> <tr> <td>22000</td> <td>Fishing Statistical area</td> <td>20000</td> </tr> <tr> <td>21001</td> <td>Inland/marine</td> <td>21000</td> </tr> <tr> <td>21002</td> <td>Ocean</td> <td>21000</td> </tr> <tr> <td>21003</td> <td>North/South/Equatorial</td> <td>21000</td> </tr> <tr> <td>22011</td> <td>FAO statistical area</td> <td>22000</td> </tr> <tr> <td>22012</td> <td>Areal grid system</td> <td>22000</td> </tr> </tbody> </table>	ID	CSI Name	Parent ID	20000	Water area		21000	Environmental area	20000	24020	Jurisdiction area	20000	22000	Fishing Statistical area	20000	21001	Inland/marine	21000	21002	Ocean	21000	21003	North/South/Equatorial	21000	22011	FAO statistical area	22000	22012	Areal grid system	22000
ID	CSI Name	Parent ID																													
20000	Water area																														
21000	Environmental area	20000																													
24020	Jurisdiction area	20000																													
22000	Fishing Statistical area	20000																													
21001	Inland/marine	21000																													
21002	Ocean	21000																													
21003	North/South/Equatorial	21000																													
22011	FAO statistical area	22000																													
22012	Areal grid system	22000																													
OUTPUT: Designed Ontology																															
General	The generated ontology will be based on the taxonomy architectural pattern (AP-TX-01) [SFBG ⁺ 07]. Each category in the classification scheme is mapped to a class, and the semantics of the relationship between children and parent categories are disambiguated by using an external resource. In the case of that the external resource does not provide any relation between two items, the pattern takes advantage of the use of logical patterns for asserting the relation <i>partOf</i> or <i>subClassOf</i> .																														
Graphical Representation																															

Table 4.3: Pattern for re-engineering a classification scheme, which follows the adjacency list data model, into an ontology schema(continued)

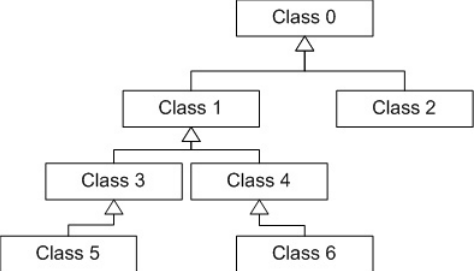
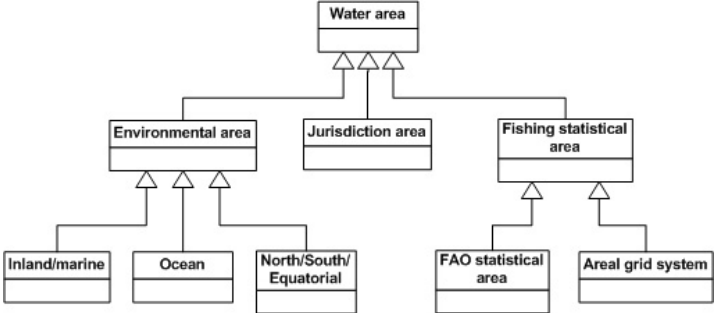
Slot	Value
(UML) General Solution Ontology	 <pre> classDiagram class Class0 class Class1 class Class2 class Class3 class Class4 class Class5 class Class6 Class0 < -- Class1 Class0 < -- Class2 Class1 < -- Class3 Class1 < -- Class4 Class3 < -- Class5 Class4 < -- Class6 </pre>
(UML) Example Solution Ontology	 <pre> classDiagram class WaterArea[Water area] class EnvironmentalArea[Environmental area] class JurisdictionArea[Jurisdiction area] class FishingStatisticalArea[Fishing statistical area] class InlandMarine[Inland/marine] class Ocean[Ocean] class NorthSouthEquatorial[North/South/Equatorial] class FAOStatisticalArea[FAO statistical area] class ArealGridSystem[Areal grid system] WaterArea < -- EnvironmentalArea WaterArea < -- JurisdictionArea WaterArea < -- FishingStatisticalArea EnvironmentalArea < -- InlandMarine EnvironmentalArea < -- Ocean EnvironmentalArea < -- NorthSouthEquatorial FishingStatisticalArea < -- FAOStatisticalArea FishingStatisticalArea < -- ArealGridSystem </pre>
PROCESS: How to Re-engineer	
General	<ol style="list-style-type: none"> 1. Identify the classification scheme items which do not have a parent key value, i.e. classification scheme items without parents. 2. For each one of the above identified classification scheme items ce_i: <ol style="list-style-type: none"> 2.1. Create the corresponding ontology class, C_i class. 2.2. Identify the classification scheme items, ce_j, which are children of ce_i, by using the parent key values. 2.3. For each one of the above identified classification scheme items ce_j: <ol style="list-style-type: none"> 2.3.1. Create the corresponding ontology class, C_j class. 2.3.2. Using the external resource identify the semantics of the relation between C_j and C_i, and set up the relation identified. 2.3.3. Repeat from step 2.2 for ce_j as a new ce_i. 3. If there are more than one classification scheme items without parent ce_i <ol style="list-style-type: none"> 3.1. Create an <i>ad-hoc</i> class as the root class of the ontology. 3.2. Using the external resource identify the semantics of the relation between C_i class and the root class, and set up the relation identified.

Table 4.3: Pattern for re-engineering a classification scheme, which follows the adjacency list data model, into an ontology schema(continued)

Slot	Value
Example	<ol style="list-style-type: none"> 1. Create the <code>Water</code> area class. 2. Create the <code>Environmental</code> area class. 3. Using the external resource identify the semantics of the relation between <code>Environmental</code> area class and the <code>Water</code> area class, and set up the relation identified. <ol style="list-style-type: none"> 3.1. Create the <code>Inland/marine</code> class. 3.2. Using the external resource identify the semantics of the relation between <code>Inland/marine</code> class and the <code>Environmental</code> area class, and set up the relation identified. 3.3. Create the <code>Ocean</code> class. 3.4. Using the external resource identify the semantics of the relation between <code>Ocean</code> class and the <code>Environmental</code> area class, and set up the relation identified. 3.5. Create the <code>North/South/Equatorial</code> class. 3.6. Using the external resource identify the semantics of the relation between <code>North/South/Equatorial</code> class and the <code>Environmental</code> area class, and set up the relation identified. 4. Create the <code>Fishing Statistical</code> area class. 5. Using the external resource identify the semantics of the relation between <code>Fishing Statistical</code> area class and the <code>Water</code> area class, and set up the relation identified. <ol style="list-style-type: none"> 5.1. Create the <code>FAO statistical</code> area class. 5.2. Using the external resource identify the semantics of the relation between <code>FAO statistical</code> area class and the <code>Fishing Statistical</code> area class, and set up the relation identified. 5.3. Create the <code>Areal grid system</code> class. 5.4. Using the external resource identify the semantics of the relation between <code>Areal grid system</code> class and the <code>Fishing Statistical</code> area class, and set up the relation identified. 6. Create the <code>Jurisdiction</code> area class 7. Using the external resource identify the semantics of the relation between <code>Jurisdiction</code> area class and the <code>Water</code> area class, and set up the relation identified.
Formal Transformation	
General	Classification Scheme: $CS = \langle CSS, CSD \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $CSD \rightarrow OS$
Relationships	
Relations to other modelling components	Use the Architectural Pattern: TX-AP-01 [SFBG ⁺ 07]

PR-NOR-CLTX-03. Pattern for re-engineering a classification scheme, which follows the snowflake data model, into an ontology schema. The pattern for re-engineering non-ontological resource shown in Table 4.4 suggests a guide to transform a classification scheme into an ontology schema. The classification scheme is modeled/represented with a snowflake data model.

Table 4.4: Pattern for re-engineering a classification scheme, which follows the snowflake data model, into an ontology schema.

Slot	Value
General Information	

Table 4.4: Pattern for re-engineering a classification scheme, which follows the snowflake data model, into an ontology schema(continued)

Slot	Value																																																							
Name	Pattern for Re-engineering a Classification Scheme, which follows the Snowflake Data Model, into an Ontology Schema																																																							
Identifier	PR-NOR-CLTX-03																																																							
Type of Component	Pattern for Re-engineering Non-Ontological Resource (PR-NOR)																																																							
Use Case																																																								
General	Re-engineering a classification scheme, which follows the snowflake model, to design an ontology schema.																																																							
Example	Suppose that someone wants to build an ontology based on an occupation hierarchical classification, which follows the snowflake data model.																																																							
Pattern for Re-engineering Non-Ontological Resource																																																								
INPUT: Resource to be Re-engineered																																																								
General	A non-ontological resource holds a classification scheme which follows the snowflake model. A classification scheme is a rooted tree of concepts, in which each concept groups entities by some particular degree of similarity. The semantics of the hierarchical relation between parents and children concepts may vary depending of the context. The snowflake data model [MZ06] is a normalized structure for hierarchy representations. In this case, the classification scheme items are grouped by levels or entities. There are as many groups as levels the classification scheme has.																																																							
Example	Snowflake models are widely used on data warehouses to build hierarchical classifications on structures known as dimensions. Some examples of dimension are Time, Product Category, Geography, Occupations, etc. In this pattern the example is an occupation hierarchical classification hold on four different tables, one for each level (PROFESSIONI_0, PROFESSIONI_1, PROFESSIONI_2, PROFESSIONI_3).																																																							
Graphical Representation																																																								
General	<table border="1" style="margin-bottom: 10px;"> <thead> <tr> <th colspan="3">First level cs items entity</th> </tr> <tr> <th>ID</th> <th>CS Name</th> <th>CS Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>CSItem1 Level 1</td> <td>CSItem1 Level 1 Desc</td> </tr> <tr> <td>2</td> <td>CSItem2 Level 1</td> <td>CSItem2 Level 1 Desc</td> </tr> <tr> <td>...</td> <td>...</td> <td>...</td> </tr> </tbody> </table> <table border="1" style="margin-bottom: 10px;"> <thead> <tr> <th colspan="4">Second level cs items entity</th> </tr> <tr> <th>ID</th> <th>First level cs item</th> <th>CS Name</th> <th>CS Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>CSItem1 Level 2</td> <td>CSItem1 Level 2 Desc</td> </tr> <tr> <td>2</td> <td>1</td> <td>CSItem2 Level 2</td> <td>CSItem2 Level 2 Desc</td> </tr> <tr> <td>...</td> <td>..</td> <td>...</td> <td>...</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th colspan="4">Third level cs items entity</th> </tr> <tr> <th>ID</th> <th>Second level cs item</th> <th>CS Name</th> <th>CS Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>CSItem1 Level 3</td> <td>CSItem1 Level 3 Desc</td> </tr> <tr> <td>2</td> <td>2</td> <td>CSItem2 Level 3</td> <td>CSItem2 Level 3 Desc</td> </tr> <tr> <td>...</td> <td>..</td> <td>...</td> <td>...</td> </tr> </tbody> </table>	First level cs items entity			ID	CS Name	CS Description	1	CSItem1 Level 1	CSItem1 Level 1 Desc	2	CSItem2 Level 1	CSItem2 Level 1 Desc	Second level cs items entity				ID	First level cs item	CS Name	CS Description	1	1	CSItem1 Level 2	CSItem1 Level 2 Desc	2	1	CSItem2 Level 2	CSItem2 Level 2 Desc	Third level cs items entity				ID	Second level cs item	CS Name	CS Description	1	1	CSItem1 Level 3	CSItem1 Level 3 Desc	2	2	CSItem2 Level 3	CSItem2 Level 3 Desc
First level cs items entity																																																								
ID	CS Name	CS Description																																																						
1	CSItem1 Level 1	CSItem1 Level 1 Desc																																																						
2	CSItem2 Level 1	CSItem2 Level 1 Desc																																																						
...																																																						
Second level cs items entity																																																								
ID	First level cs item	CS Name	CS Description																																																					
1	1	CSItem1 Level 2	CSItem1 Level 2 Desc																																																					
2	1	CSItem2 Level 2	CSItem2 Level 2 Desc																																																					
...																																																					
Third level cs items entity																																																								
ID	Second level cs item	CS Name	CS Description																																																					
1	1	CSItem1 Level 3	CSItem1 Level 3 Desc																																																					
2	2	CSItem2 Level 3	CSItem2 Level 3 Desc																																																					
...																																																					
Example	<table border="1" style="margin-bottom: 10px;"> <thead> <tr> <th colspan="2">PROFESSIONE 0</th> </tr> <tr> <th>ID 0</th> <th>Desc 0</th> </tr> </thead> <tbody> <tr> <td>01</td> <td>Professioni specialistiche e tecniche</td> </tr> <tr> <td>02</td> <td>Professioni operative della gestione dimpresa</td> </tr> <tr> <td>...</td> <td>...</td> </tr> </tbody> </table> <table border="1" style="margin-bottom: 10px;"> <thead> <tr> <th colspan="3">PROFESSIONE 1</th> </tr> <tr> <th>ID 1</th> <th>Desc 1</th> <th>ID 0</th> </tr> </thead> <tbody> <tr> <td>01.05</td> <td>Specialist e tecnici delle scienze informatiche</td> <td>01</td> </tr> <tr> <td>02.05</td> <td>Specialist e tecnici delle gestione dimpresa</td> <td>02</td> </tr> <tr> <td>...</td> <td>...</td> <td>...</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th colspan="3">PROFESSIONE 2</th> </tr> <tr> <th>ID 2</th> <th>Desc 2</th> <th>ID 1</th> </tr> </thead> <tbody> <tr> <td>01.05.01</td> <td>Specialist e tecnici delle scienze informatiche</td> <td>01</td> </tr> <tr> <td>02.05.01</td> <td>Specialist e tecnici delle gestione dimpresa</td> <td>02</td> </tr> <tr> <td>...</td> <td>...</td> <td>...</td> </tr> </tbody> </table>	PROFESSIONE 0		ID 0	Desc 0	01	Professioni specialistiche e tecniche	02	Professioni operative della gestione dimpresa	PROFESSIONE 1			ID 1	Desc 1	ID 0	01.05	Specialist e tecnici delle scienze informatiche	01	02.05	Specialist e tecnici delle gestione dimpresa	02	PROFESSIONE 2			ID 2	Desc 2	ID 1	01.05.01	Specialist e tecnici delle scienze informatiche	01	02.05.01	Specialist e tecnici delle gestione dimpresa	02															
PROFESSIONE 0																																																								
ID 0	Desc 0																																																							
01	Professioni specialistiche e tecniche																																																							
02	Professioni operative della gestione dimpresa																																																							
...	...																																																							
PROFESSIONE 1																																																								
ID 1	Desc 1	ID 0																																																						
01.05	Specialist e tecnici delle scienze informatiche	01																																																						
02.05	Specialist e tecnici delle gestione dimpresa	02																																																						
...																																																						
PROFESSIONE 2																																																								
ID 2	Desc 2	ID 1																																																						
01.05.01	Specialist e tecnici delle scienze informatiche	01																																																						
02.05.01	Specialist e tecnici delle gestione dimpresa	02																																																						
...																																																						

Table 4.4: Pattern for re-engineering a classification scheme, which follows the snowflake data model, into an ontology schema(continued)

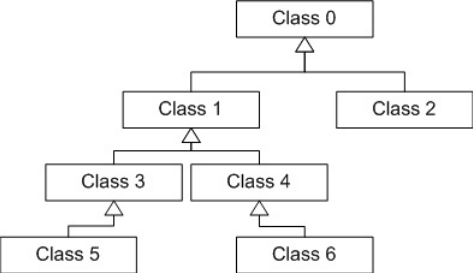
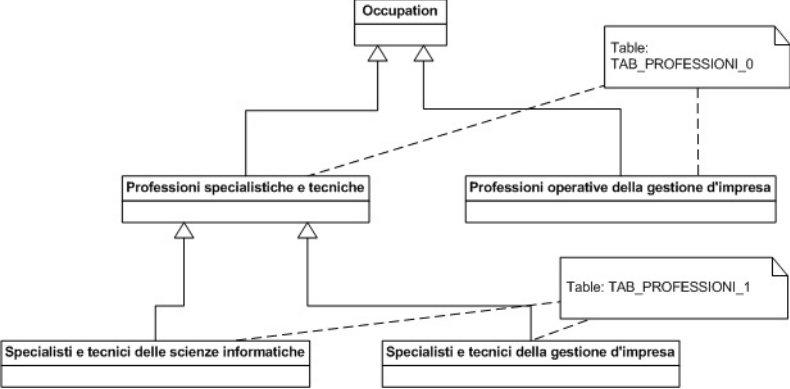
Slot	Value
OUTPUT: Designed Ontology	
General	<p>The generated ontology will be based on the taxonomy architectural pattern (AP-TX-01) [SFBG⁺07]. Each category in the classification scheme is mapped to a class, and the semantics of the relationship between children and parent categories are disambiguated by using an external resource. In the case of that the external resource does not provide any relation between two items, the pattern takes advantage of the use of logical patterns for asserting the relation <i>partOf</i> or <i>subClassOf</i>.</p>
Graphical Representation	
(UML) General Solution Ontology	 <pre> classDiagram Class0 < -- Class1 Class0 < -- Class2 Class1 < -- Class3 Class1 < -- Class4 Class3 < -- Class5 Class4 < -- Class6 </pre>
(UML) Example Solution Ontology	
PROCESS: How to Re-engineer	
General	<ol style="list-style-type: none"> 1. Select all the classification scheme items from the first level. 2. For each one of the above selected classification scheme items ce_i: <ol style="list-style-type: none"> 2.1. Create the corresponding ontology class, C_i class. 2.2. Identify the classification scheme items, ce_j, on the next level, which are children of ce_i, by using the parent key values. 2.3. For each one of the above identified classification scheme items ce_j: <ol style="list-style-type: none"> 2.3.1. Create the corresponding ontology class, C_j class. 2.3.2. Using the external resource identify the semantics of the relation between C_j and C_i, and set up the relation identified. 2.3.3. Repeat from step 2.2 for ce_j as a new ce_i. 3. If there are more than one classification scheme items from the first level ce_i <ol style="list-style-type: none"> 3.1. Create an <i>ad-hoc</i> class as the root class of the ontology. 3.2. Using the external resource identify the semantics of the relation between C_i class and the root class, and set up the relation identified.

Table 4.4: Pattern for re-engineering a classification scheme, which follows the snowflake data model, into an ontology schema(continued)

Slot	Value
Example	<ol style="list-style-type: none"> 1. Create the <i>Professioni specialistiche e tecniche</i> class. <ol style="list-style-type: none"> 1.1. Create the <i>Specialist e tecnici delle scienze informatiche</i> class. 1.2. Using the external resource identify the semantics of the relation between the <i>Specialist e tecnici delle scienze informatiche</i> class and the <i>Professioni specialistiche e tecniche</i> class, and set up the relation identified. 1.3. Create the <i>Specialist e tecnici delle gestione dimpresa</i> class. 1.4. Using the external resource identify the semantics of the relation between the <i>Specialist e tecnici delle gestione dimpresa</i> class and the <i>Professioni specialistiche e tecniche</i> class, and set up the relation identified. 2. Create the <i>Professioni operative della gestione dimpresa</i> class. 3. Create the <i>Occupation</i> class. 4. Using the external resource identify the semantics of the relation between the <i>Professioni specialistiche e tecniche</i> class and the <i>Occupation</i> class, and set up the relation identified. 5. Using the external resource identify the semantics of the relation between the <i>Professioni operative della gestione dimpresa</i> class and the <i>Occupation</i> class, and set up the relation identified.
Formal Transformation	
General	Classification Scheme: $CS = \langle CSS, CSD \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $CSD \rightarrow OS$
Relationships	
Relations to other modelling components	Use the Architectural Pattern: TX-AP-01 [SFBG ⁺ 07]

PR-NOR-CLTX-04. Pattern for re-engineering a classification scheme, which follows the flattened data model, into an ontology schema. The pattern for re-engineering non-ontological resource shown in Table 4.5 suggests a guide to transform a classification scheme into an ontology schema. The classification scheme is modeled/represented with a flattened data model.

Table 4.5: Pattern for re-engineering a classification scheme, which follows the flattened data model, into an ontology schema.

Slot	Value
General Information	
Name	Pattern for Re-engineering a Classification Scheme, which follows the Flattened Data Model, into an Ontology Schema
Identifier	PR-NOR-CLTX-04
Type of Component	Pattern for Re-engineering Non-Ontological Resource (PR-NOR)
Use Case	
General	Re-engineering a classification scheme, which follows the flattened model, to design an ontology schema.
Example	Suppose that someone wants to build an ontology based on a classification published as one table with a column for each classification level.
Pattern for Re-engineering Non-Ontological Resource	
INPUT: Resource to be Re-engineered	

Table 4.5: Pattern for re-engineering a classification scheme, which follows the flattened data model, into an ontology schema(continued)

Slot	Value																																										
General	<p>A non-ontological resource holds a classification scheme which follows the flattened data model.</p> <p>A classification scheme is a rooted tree of concepts, in which each concept groups entities by some particular degree of similarity. The semantics of the hierarchical relation between parents and children concepts may vary depending of the context.</p> <p>The flattened data model [MZ06] is a denormalized structure for hierarchy representations. In this case, each hierarchy level is represented on a different column. There are as many columns as levels the classification scheme has. Therefore each row has the complete path from the root to a leaf node.</p>																																										
Example	<p>The Classification of Italian Education Titles published by the National Institute of Statistics (ISTAT) is represented following a flattened model. The first level of the classification (level code) is related to the education title level which comprises values as elementary, media, university, master, etc. The second level of the classification is the type of school or institute which offers the education title. The last level is the education title itself; it has a specific specialization code and also a code which is the concatenation of the previous code levels.</p>																																										
Graphical Representation																																											
General	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="6" style="background-color: #cccccc;">Flattened Entity</th> </tr> <tr> <th colspan="2" style="background-color: #cccccc;">First level</th> <th colspan="2" style="background-color: #cccccc;">Second level</th> <th colspan="2" style="background-color: #cccccc;">Third level</th> <th style="background-color: #cccccc;">...</th> </tr> <tr> <th style="background-color: #cccccc;">ID</th> <th style="background-color: #cccccc;">CS Name</th> <th style="background-color: #cccccc;">ID</th> <th style="background-color: #cccccc;">CS Name</th> <th style="background-color: #cccccc;">ID</th> <th style="background-color: #cccccc;">CS Name</th> <th style="background-color: #cccccc;">...</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>CSItem1 Level 1</td> <td>1</td> <td>CSItem1 Level 2</td> <td>1</td> <td>CSItem1 Level 3</td> <td>...</td> </tr> <tr> <td>2</td> <td>CSItem2 Level 1</td> <td>2</td> <td>CSItem2 Level 2</td> <td>2</td> <td>CSItem2 Level 3</td> <td>...</td> </tr> <tr> <td>...</td> <td>...</td> <td>...</td> <td>...</td> <td>...</td> <td>...</td> <td>..</td> </tr> </tbody> </table>	Flattened Entity						First level		Second level		Third level		...	ID	CS Name	ID	CS Name	ID	CS Name	...	1	CSItem1 Level 1	1	CSItem1 Level 2	1	CSItem1 Level 3	...	2	CSItem2 Level 1	2	CSItem2 Level 2	2	CSItem2 Level 3	
Flattened Entity																																											
First level		Second level		Third level		...																																					
ID	CS Name	ID	CS Name	ID	CS Name	...																																					
1	CSItem1 Level 1	1	CSItem1 Level 2	1	CSItem1 Level 3	...																																					
2	CSItem2 Level 1	2	CSItem2 Level 2	2	CSItem2 Level 3	...																																					
...																																					
Example	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="7" style="background-color: #cccccc;">ISTAT - CTS103 - Italian Education Titles Classification</th> </tr> <tr> <th style="background-color: #cccccc;">ID Education</th> <th style="background-color: #cccccc;">Code of school or academic course specialization</th> <th style="background-color: #cccccc;">Education Title</th> <th style="background-color: #cccccc;">Code of school, institute or group.</th> <th style="background-color: #cccccc;">Type of school, institute or group</th> <th style="background-color: #cccccc;">Level ID</th> <th style="background-color: #cccccc;">...</th> </tr> </thead> <tbody> <tr> <td>30101010</td> <td>010</td> <td>Esperto frutticoltore</td> <td>101</td> <td>Istituto professionale agrario</td> <td>30</td> <td></td> </tr> <tr> <td>30101011</td> <td>011</td> <td>Esperto olivicoltore</td> <td>101</td> <td>Istituto professionale agrario</td> <td>30</td> <td></td> </tr> <tr> <td>40104001</td> <td>001</td> <td>Analista contabile</td> <td>104</td> <td>Istit. Profess.Servizi Commerciali</td> <td>40</td> <td></td> </tr> <tr> <td>40104002</td> <td>002</td> <td>Operatore commerciale</td> <td>104</td> <td>Istit. Profess.Servizi Commerciali</td> <td>40</td> <td></td> </tr> </tbody> </table>	ISTAT - CTS103 - Italian Education Titles Classification							ID Education	Code of school or academic course specialization	Education Title	Code of school, institute or group.	Type of school, institute or group	Level ID	...	30101010	010	Esperto frutticoltore	101	Istituto professionale agrario	30		30101011	011	Esperto olivicoltore	101	Istituto professionale agrario	30		40104001	001	Analista contabile	104	Istit. Profess.Servizi Commerciali	40		40104002	002	Operatore commerciale	104	Istit. Profess.Servizi Commerciali	40	
ISTAT - CTS103 - Italian Education Titles Classification																																											
ID Education	Code of school or academic course specialization	Education Title	Code of school, institute or group.	Type of school, institute or group	Level ID	...																																					
30101010	010	Esperto frutticoltore	101	Istituto professionale agrario	30																																						
30101011	011	Esperto olivicoltore	101	Istituto professionale agrario	30																																						
40104001	001	Analista contabile	104	Istit. Profess.Servizi Commerciali	40																																						
40104002	002	Operatore commerciale	104	Istit. Profess.Servizi Commerciali	40																																						
OUTPUT: Designed Ontology																																											
General	<p>The generated ontology will be based on the taxonomy architectural pattern (AP-TX-01) [SFBG⁺07]. Each category in the classification scheme is mapped to a class, and the semantics of the relationship between children and parent categories are disambiguated by using an external resource. In the case of that the external resource does not provide any relation between two items, the pattern takes advantage of the use of logical patterns for asserting the relation <i>partOf</i> or <i>subClassOf</i>.</p>																																										
Graphical Representation																																											
(UML) General Solution Ontology	<pre> classDiagram class Class0 class Class1 class Class2 class Class3 class Class4 class Class5 class Class6 Class0 < -- Class1 Class0 < -- Class2 Class1 < -- Class3 Class1 < -- Class4 Class3 < -- Class5 Class4 < -- Class6 </pre>																																										

Table 4.5: Pattern for re-engineering a classification scheme, which follows the flattened data model, into an ontology schema(continued)

Slot	Value
<p>(UML) Example Solution Ontology</p>	<pre> classDiagram class EducationTitles[Education Titles] class HSE[HIGHER SECONDARY EDUCATION] class HSE_AATU[HIGHER SECONDARY EDUCATION - ALLOWS ACCESS TO UNIVERSITIES] class IPA[Istituto professionale agrario] class IPC[Istit. profess. servizi commerciali...] class EFC[Esperto frutticoltore] class EOC[Esperto olivicoltore] class AC[Analista contabile] class OC[Operatore commerciale] EducationTitles < -- HSE EducationTitles < -- HSE_AATU HSE < -- IPA HSE < -- IPC IPA < -- EFC IPA < -- EOC IPC < -- AC IPC < -- OC </pre>
<p>PROCESS: How to Re-engineer</p>	
<p>General</p>	<ol style="list-style-type: none"> 1. Select all the classification scheme items from the first level, using the level column and taking care to avoid duplicity. 2. For each one of the above selected classification scheme items ce_i: <ol style="list-style-type: none"> 2.1. Create the corresponding ontology class, C_i class. 2.2. Identify the classification scheme items, ce_j, on the next level, which are children of ce_i, by using the path and level columns. 2.3. For each one of the above identified classification scheme items ce_j: <ol style="list-style-type: none"> 2.3.1. Create the corresponding ontology class, C_j class. 2.3.2. Using the external resource identify the semantics of the relation between C_j and C_i, and set up the relation identified. 2.3.3. Repeat from step 2.2 for ce_j as a new ce_i. 3. If there are more than one classification scheme items from the first level ce_i <ol style="list-style-type: none"> 3.1. Create an <i>ad-hoc</i> class as the root class of the ontology. 3.2. Using the external resource identify the semantics of the relation between C_i class and the root class, and set up the relation identified.
<p>Example</p>	<ol style="list-style-type: none"> 1. Create the HIGHER SECONDARY EDUCATION class. <ol style="list-style-type: none"> 1.1. Create the Istituto professionale agrario class. 1.2. Using the external resource identify the semantics of the relation between the Istituto professionale agrario class and the HIGHER SECONDARY EDUCATION class, and set up the relation identified. <ol style="list-style-type: none"> 1.2.1. Create the Esperto frutticoltore class. 1.2.2. Using the external resource identify the semantics of the relation between the Esperto frutticoltore class and the Istituto professionale agrario class, and set up the relation identified. 1.2.3. Create the Esperto olivicoltore class. 1.2.4. Using the external resource identify the semantics of the relation between the Esperto olivicoltore class and the Istituto professionale agrario class, and set up the relation identified. 2. Create the HIGHER SECONDARY EDUCATION - ALLOWS ACCESS TO UNIVERSITIES class. 3. Create the Education Title class. 4. Using the external resource identify the semantics of the relation between the HIGHER SECONDARY EDUCATION class and the Education Title class, and set up the relation identified. 5. Using the external resource identify the semantics of the relation between the HIGHER SECONDARY EDUCATION - ALLOWS ACCESS TO UNIVERSITIES class and the Education Title class, and set up the relation identified.

Table 4.5: Pattern for re-engineering a classification scheme, which follows the flattened data model, into an ontology schema(continued)

Slot	Value
Formal Transformation	
General	Classification Scheme: $CS = \langle CSS, CSD \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $CSD \rightarrow OS$
Relationships	
Relations to other modelling components	Use the Architectural Pattern: TX-AP-01 [SFBG ⁺ 07]

Patterns for the ABox transformations

These patterns transform the resource schema into an ontology schema, and the resource content, into ontology instances. The ABox transformation approach leaves the informal semantics of the re-engineered resources mostly untouched [SAd⁺07a].

The patterns presented here deal with classification schemes. According to [VTAGS⁺08], the schema of a classification scheme has the following components: (1) classification scheme item, which will be transformed to a class, and (2) classification scheme item relationship, which will be transformed either to a *subClassOf* or a *partOf* relation. The patterns take advantage of the use of logical patterns⁸ for asserting the *partOf* or *subClassOf* relation. Finally, the content of the classification scheme will be transformed into ontology instances.

PR-NOR-CLLO-10. Pattern for re-engineering a classification scheme, which follows the path enumeration data model, into an ontology. The pattern for re-engineering non-ontological resource shown in Table 4.6 suggests a guide to transform a classification scheme, which follows the path enumeration data model, into an ontology. The pattern transforms the resource schema into an ontology schema, and the resource content, into ontology instances.

Table 4.6: Pattern for re-engineering a classification scheme, which follows the path enumeration data model, into an ontology.

Slot	Value
General Information	
Name	Pattern for Re-engineering a Classification Scheme, which follows the Path Enumeration Data Model, into an Ontology Schema.
Identifier	PR-NOR-CLLO-10
Type of Component	Pattern for Re-engineering Non-Ontological Resource (PR-NOR)
Use Case	
General	Re-engineering a classification scheme, which follows the path enumeration model, to design an ontology schema.
Example	Suppose that someone wants to build an ontology based on the International Standard Classification of Occupations (for European Union purposes) ISCO-88 (COM). This classification scheme follows the path enumeration data model.
Pattern for Re-engineering Non-Ontological Resource	
INPUT: Resource to be Re-engineered	

⁸Logical patterns are included in the ODP portal at <http://ontologydesignpatterns.org>

Table 4.6: Pattern for re-engineering a classification scheme, which follows the path enumeration data model, into an ontology.(continued)

Slot	Value																					
General	<p>A non-ontological resource holds a classification scheme which follows the path enumeration model.</p> <p>A classification scheme is a rooted tree of concepts, in which each concept groups entities by some particular degree of similarity. The semantics of the hierarchical relation between parents and children concepts may vary depending of the context.</p> <p>The path enumeration data model [Bra05], for classification schemes, takes advantage of that there is one and only one path from the root to every item in the classification. The path enumeration model stores that path as string by concatenating either the edges or the keys of the classification scheme items in the path.</p>																					
Example	<p>The International Standard Classification of Occupations (for European Union purposes), 1988 version: ISCO-88 (COM) published by Eurostat is modeled with the path enumeration data model. This classification scheme is available at http://ec.europa.eu/eurostat/ramon/</p>																					
Graphical Representation																						
General	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr style="background-color: black; color: white;"> <th>Path Enumeration</th> <th>CS Name</th> <th>CS Description</th> </tr> </thead> <tbody> <tr><td>1</td><td>Name1</td><td>Description1</td></tr> <tr><td>11</td><td>Name11</td><td>Description11</td></tr> <tr><td>111</td><td>Name111</td><td>Description111</td></tr> <tr><td>12</td><td>Name12</td><td>Description12</td></tr> <tr><td>2</td><td>Name2</td><td>Description2</td></tr> <tr><td>...</td><td>...</td><td>...</td></tr> </tbody> </table>	Path Enumeration	CS Name	CS Description	1	Name1	Description1	11	Name11	Description11	111	Name111	Description111	12	Name12	Description12	2	Name2	Description2
Path Enumeration	CS Name	CS Description																				
1	Name1	Description1																				
11	Name11	Description11																				
111	Name111	Description111																				
12	Name12	Description12																				
2	Name2	Description2																				
...																				
Example	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr style="background-color: black; color: white;"> <th>Code</th> <th>Level</th> <th>Name</th> </tr> </thead> <tbody> <tr><td>1</td><td>1</td><td>LEGISLATORS, SENIOR OFFICIALS AND MANAGERS</td></tr> <tr><td>11</td><td>2</td><td>Legislators and senior officials</td></tr> <tr><td>111</td><td>3</td><td>Corporate managers</td></tr> <tr><td>2</td><td>1</td><td>PROFESSIONALS</td></tr> <tr><td>...</td><td>...</td><td>...</td></tr> </tbody> </table>	Code	Level	Name	1	1	LEGISLATORS, SENIOR OFFICIALS AND MANAGERS	11	2	Legislators and senior officials	111	3	Corporate managers	2	1	PROFESSIONALS			
Code	Level	Name																				
1	1	LEGISLATORS, SENIOR OFFICIALS AND MANAGERS																				
11	2	Legislators and senior officials																				
111	3	Corporate managers																				
2	1	PROFESSIONALS																				
...																				
OUTPUT: Designed Ontology																						
General	<p>The ontology generated will be based on the taxonomy architectural pattern (AP-TX-01) [SFBG⁺07]. The classification scheme item will be transformed to a class. The classification scheme item relationship will be transformed either to a <i>subClassOf</i> or a <i>partOf</i> relation. The pattern takes advantage of the use of logical patterns for asserting the <i>partOf</i> or <i>subClassOf</i> relation. Finally, the content of the classification scheme will be transformed into ontology instances.</p>																					
Graphical Representation																						
(UML) General Solution Ontology	<pre> classDiagram class Class0 class Instance1 class Instance2 class Instance3 Class0 -- > Instance1 Class0 -- > Instance2 Class0 -- > Instance3 Class0 --> Instance1 : type Class0 --> Instance2 : type Class0 --> Instance3 : type </pre>																					
(UML) Example Solution Ontology	<pre> classDiagram class Occupation class PROFESSIONALS class LEGISLATORS_SENIOR_OFFICIALS_AND_MANAGERS class Corporate_managers class Legislators_and_senior_officials Occupation -- > PROFESSIONALS Occupation -- > LEGISLATORS_SENIOR_OFFICIALS_AND_MANAGERS Occupation -- > Corporate_managers Occupation -- > Legislators_and_senior_officials Occupation --> PROFESSIONALS : type Occupation --> LEGISLATORS_SENIOR_OFFICIALS_AND_MANAGERS : type Occupation --> Corporate_managers : type Occupation --> Legislators_and_senior_officials : type </pre>																					

Table 4.6: Pattern for re-engineering a classification scheme, which follows the path enumeration data model, into an ontology.(continued)

Slot	Value
PROCESS: How to Re-engineer	
General	<ol style="list-style-type: none"> 1. Create a class for the <i>classification scheme item</i> schema component. 2. Assert that the <i>classification scheme item</i> class is <i>subClassOf</i> itself, by using the proper logical pattern. 3. Select all the classification scheme items, by using the path enumeration data model information. 4. For each one of the classification scheme items ce_i: <ol style="list-style-type: none"> 4.1. Create the corresponding ontology instance ice_i. 4.2. Assert that ice_i is instance of the <i>classification scheme item</i> class.
Example	<ol style="list-style-type: none"> 1. Create a <i>Occupation</i> class for the classification scheme item schema component. 2. Assert that the <i>Occupation</i> class is <i>subClassOf</i> itself, by using the proper logical pattern. <ol style="list-style-type: none"> 2.1. Create the <i>LEGISLATORS, SENIOR OFFICIALS AND MANAGERS</i> instance and assert that it is instance of the <i>Occupation</i> class. 2.2. Create the <i>Legislators</i> and <i>senior officials</i> instance and assert that it is instance of the <i>Occupation</i> class. 2.3. Create the <i>Corporate managers</i> instance and assert that it is instance of the <i>Occupation</i> class. 2.4. Create the <i>PROFESSIONALS</i> instance and assert that it is instance of the <i>Occupation</i> class.
Formal Transformation	
General	Classification Scheme: $CS = \langle CSS, CSD \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $CSS \rightarrow OS$ $CSD \rightarrow KB$
Relationships	
Relations to other modelling components	Use the Architectural Pattern: TX-AP-01 [SFBG ⁺ 07]

PR-NOR-CLLO-11. Pattern for re-engineering a classification scheme, which follows the adjacency list data model, into an ontology. The pattern for re-engineering non-ontological resource shown in Table 4.7 suggests a guide to transform a classification scheme, which follows the adjacency list data model, into an ontology. The pattern transforms the resource schema into an ontology schema, and the resource content, into ontology instances.

Table 4.7: Pattern for re-engineering a classification scheme, which follows the adjacency list data model, into an ontology.

Slot	Value
General Information	
Name	Pattern for Re-engineering a Classification Scheme, which follows the Adjacency List Data Model, into an Ontology
Identifier	PR-NOR-CLLO-11
Type of Component	Pattern for Re-engineering Non-Ontological Resource (PR-NOR)
Use Case	
General	Re-engineering a classification scheme, which follows the adjacency list model, to design an ontology.

Table 4.7: Pattern for re-engineering a classification scheme, which follows the adjacency list data model, into an ontology (continued)

Slot	Value																														
Example	Suppose that someone wants to build an ontology based on the water areas classification published by FAO. This classification scheme follows the adjacency list data model.																														
Pattern for Re-engineering Non-Ontological Resource																															
INPUT: Resource to be Re-engineered																															
General	<p>A non-ontological resource holds a classification scheme which follows the adjacency list model.</p> <p>A classification scheme is a rooted tree of concepts, in which each concept groups entities by some particular degree of similarity.</p> <p>The semantics of the hierarchical relation between parents and children concepts may vary depending of the context.</p> <p>The adjacency list data model [Bra05] for hierarchical classifications proposes to create an entity which holds a list of items with a linking column associated to their parent items.</p>																														
Example	<p>The FAO classification for water areas groups them according to some different criteria as environment, statistics, and jurisdiction, among others.</p> <p>This classification scheme is available at http://www.fao.org/figis/servlet/RefServlet</p>																														
Graphical Representation																															
General	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>ID</th> <th>CS Name</th> <th>Parent ID</th> </tr> </thead> <tbody> <tr><td>1</td><td>Category1</td><td>Null</td></tr> <tr><td>2</td><td>Category2</td><td>Null</td></tr> <tr><td>3</td><td>Category3</td><td>1</td></tr> <tr><td>4</td><td>Category4</td><td>1</td></tr> <tr><td>5</td><td>Category6</td><td>3</td></tr> <tr><td>6</td><td>Category7</td><td>4</td></tr> <tr><td>...</td><td>...</td><td>...</td></tr> </tbody> </table>	ID	CS Name	Parent ID	1	Category1	Null	2	Category2	Null	3	Category3	1	4	Category4	1	5	Category6	3	6	Category7	4						
ID	CS Name	Parent ID																													
1	Category1	Null																													
2	Category2	Null																													
3	Category3	1																													
4	Category4	1																													
5	Category6	3																													
6	Category7	4																													
...																													
Example	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>ID</th> <th>CSI Name</th> <th>Parent ID</th> </tr> </thead> <tbody> <tr><td>20000</td><td>Water area</td><td></td></tr> <tr><td>21000</td><td>Environmental area</td><td>20000</td></tr> <tr><td>24020</td><td>Jurisdiction area</td><td>20000</td></tr> <tr><td>22000</td><td>Fishing Statistical area</td><td>20000</td></tr> <tr><td>21001</td><td>Inland/marine</td><td>21000</td></tr> <tr><td>21002</td><td>Ocean</td><td>21000</td></tr> <tr><td>21003</td><td>North/South/Equatorial</td><td>21000</td></tr> <tr><td>22011</td><td>FAO statistical area</td><td>22000</td></tr> <tr><td>22012</td><td>Areal grid system</td><td>22000</td></tr> </tbody> </table>	ID	CSI Name	Parent ID	20000	Water area		21000	Environmental area	20000	24020	Jurisdiction area	20000	22000	Fishing Statistical area	20000	21001	Inland/marine	21000	21002	Ocean	21000	21003	North/South/Equatorial	21000	22011	FAO statistical area	22000	22012	Areal grid system	22000
ID	CSI Name	Parent ID																													
20000	Water area																														
21000	Environmental area	20000																													
24020	Jurisdiction area	20000																													
22000	Fishing Statistical area	20000																													
21001	Inland/marine	21000																													
21002	Ocean	21000																													
21003	North/South/Equatorial	21000																													
22011	FAO statistical area	22000																													
22012	Areal grid system	22000																													
OUTPUT: Designed Ontology																															
General	<p>The ontology generated will be based on the taxonomy architectural pattern (AP-TX-01) [SFBG⁺07]. The classification scheme item will be transformed to a class. The classification scheme item relationship will be transformed either to a <i>subClassOf</i> or a <i>partOf</i> relation. The pattern takes advantage of the use of logical patterns for asserting the <i>partOf</i> or <i>subClassOf</i> relation. Finally, the content of the classification scheme will be transformed into ontology instances.</p>																														
Graphical Representation																															
(UML) General Solution Ontology	<pre> classDiagram class Class0 class Instance1 class Instance2 class Instance3 Class0 --> Class0 Instance1 -- > Class0 : type Instance2 -- > Class0 : type Instance3 -- > Class0 : type </pre>																														

Table 4.7: Pattern for re-engineering a classification scheme, which follows the adjacency list data model, into an ontology (continued)

Slot	Value
(UML) Example Solution Ontology	<pre> classDiagram class Area class Environmental_area class Jurisdiction_area class Inland_marine class Ocean class North_South_Equatorial class Fishing_statistical_area class Areal_grid_system class FAO_statistical_area Area < -- Environmental_area Area < -- Jurisdiction_area Area < -- Inland_marine Area < -- Ocean Area < -- North_South_Equatorial Area < -- Fishing_statistical_area Area < -- FAO_statistical_area Fishing_statistical_area < -- Areal_grid_system Area --> Area : type </pre>
PROCESS: How to Re-engineer	
General	<ol style="list-style-type: none"> 1. Create a class for the <i>classification scheme item</i> schema component. 2. Assert that the <i>classification scheme item</i> class is <i>subClassOf</i> itself, by using the proper logical pattern. 3. Select all the classification scheme items, by using the adjacency list data model information. 4. For each one of the classification scheme items ce_i: <ol style="list-style-type: none"> 4.1. Create the corresponding ontology instance ice_i. 4.2. Assert that ice_i is instance of the <i>classification scheme item</i> class.
Example	<ol style="list-style-type: none"> 1. Create a <code>Area</code> class for the <i>classification scheme item</i> schema component. 2. Assert that the <code>Area</code> class is <i>subClassOf</i> itself, by using the proper logical pattern. <ol style="list-style-type: none"> 2.1. Create the <code>Water area</code> instance and assert that it is instance of the <code>Area</code> class. 2.2. Create the <code>Environmental area</code> instance and assert that it is instance of the <code>Area</code> class. 2.3. Create the <code>Inland/marine</code> instance and assert that it is instance of the <code>Area</code> class. 2.4. Create the <code>Ocean</code> instance and assert that it is instance of the <code>Area</code> class. 2.5. Create the <code>North/South/Equatorial</code> instance and assert that it is instance of the <code>Area</code> class. 2.6. Create the <code>Fishing Statistical area</code> instance and assert that it is instance of the <code>Area</code> class. 2.7. Create the <code>FAO statistical area</code> instance and assert that it is instance of the <code>Area</code> class. 2.8. Create the <code>Areal grid system</code> instance and assert that it is instance of the <code>Area</code> class. 2.9. Create the <code>Jurisdiction area</code> instance and assert that it is instance of the <code>Area</code> class.
Formal Transformation	
General	<p>Classification Scheme: $CS = \langle CSS, CSD \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $CSS \rightarrow OS$ $CSD \rightarrow KB$</p>
Relationships	
Relations to other modelling components	Use the Architectural Pattern: TX-AP-01 [SFBG ⁺ 07]

PR-NOR-CLLO-12. Pattern for re-engineering a classification scheme, which follows the snowflake data model, into an ontology. The pattern for re-engineering non-ontological resource shown in Table 4.8 suggests a guide to transform a classification scheme into an ontology. The classification scheme is modeled/represented with a snowflake data model. The pattern transforms the resource schema into an ontology schema, and the resource content, into ontology instances.

Table 4.8: Pattern for re-engineering a classification scheme, which follows the snowflake data model, into an ontology.

Slot	Value																																																							
General Information																																																								
Name	Pattern for Re-engineering a Classification Scheme, which follows the Snowflake Data Model, into an Ontology																																																							
Identifier	PR-NOR-CLLO-12																																																							
Type of Component	Pattern for Re-engineering Non-Ontological Resource (PR-NOR)																																																							
Use Case																																																								
General	Re-engineering a classification scheme, which follows the snowflake model, to design an ontology schema.																																																							
Example	Suppose that someone wants to build an ontology based on an occupation hierarchical classification, which follows the snowflake data model.																																																							
Pattern for Re-engineering Non-Ontological Resource																																																								
INPUT: Resource to be Re-engineered																																																								
General	A non-ontological resource holds a classification scheme which follows the snowflake model. A classification scheme is a rooted tree of concepts, in which each concept groups entities by some particular degree of similarity. The semantics of the hierarchical relation between parents and children concepts may vary depending of the context. The snowflake data model [MZ06] is a normalized structure for hierarchy representations. In this case, the classification scheme items are grouped by levels or entities. There are as many groups as levels the classification scheme has.																																																							
Example	Snowflake models are widely used on data warehouses to build hierarchical classifications on structures known as dimensions. Some examples of dimension are Time, Product Category, Geography, Occupations, etc. In this pattern the example is an occupation hierarchical classification hold on four different tables, one for each level (PROFESSIONI_0, PROFESSIONI_1, PROFESSIONI_2, PROFESSIONI_3).																																																							
Graphical Representation																																																								
General	<table border="1" style="margin: 10px auto;"> <thead> <tr> <th colspan="3">First level cs items entity</th> </tr> <tr> <th>ID</th> <th>CS Name</th> <th>CS Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>CSItem1 Level 1</td> <td>CSItem1 Level 1 Desc</td> </tr> <tr> <td>2</td> <td>CSItem2 Level 1</td> <td>CSItem2 Level 1 Desc</td> </tr> <tr> <td>...</td> <td>...</td> <td>...</td> </tr> </tbody> </table> <table border="1" style="margin: 10px auto;"> <thead> <tr> <th colspan="4">Second level cs items entity</th> </tr> <tr> <th>ID</th> <th>First level cs item</th> <th>CS Name</th> <th>CS Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>CSItem1 Level 2</td> <td>CSItem1 Level 2 Desc</td> </tr> <tr> <td>2</td> <td>1</td> <td>CSItem2 Level 2</td> <td>CSItem2 Level 2 Desc</td> </tr> <tr> <td>...</td> <td>..</td> <td>...</td> <td>...</td> </tr> </tbody> </table> <table border="1" style="margin: 10px auto;"> <thead> <tr> <th colspan="4">Third level cs items entity</th> </tr> <tr> <th>ID</th> <th>Second level cs item</th> <th>CS Name</th> <th>CS Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>CSItem1 Level 3</td> <td>CSItem1 Level 3 Desc</td> </tr> <tr> <td>2</td> <td>2</td> <td>CSItem2 Level 3</td> <td>CSItem2 Level 3 Desc</td> </tr> <tr> <td>...</td> <td>..</td> <td>...</td> <td>...</td> </tr> </tbody> </table>	First level cs items entity			ID	CS Name	CS Description	1	CSItem1 Level 1	CSItem1 Level 1 Desc	2	CSItem2 Level 1	CSItem2 Level 1 Desc	Second level cs items entity				ID	First level cs item	CS Name	CS Description	1	1	CSItem1 Level 2	CSItem1 Level 2 Desc	2	1	CSItem2 Level 2	CSItem2 Level 2 Desc	Third level cs items entity				ID	Second level cs item	CS Name	CS Description	1	1	CSItem1 Level 3	CSItem1 Level 3 Desc	2	2	CSItem2 Level 3	CSItem2 Level 3 Desc
First level cs items entity																																																								
ID	CS Name	CS Description																																																						
1	CSItem1 Level 1	CSItem1 Level 1 Desc																																																						
2	CSItem2 Level 1	CSItem2 Level 1 Desc																																																						
...																																																						
Second level cs items entity																																																								
ID	First level cs item	CS Name	CS Description																																																					
1	1	CSItem1 Level 2	CSItem1 Level 2 Desc																																																					
2	1	CSItem2 Level 2	CSItem2 Level 2 Desc																																																					
...																																																					
Third level cs items entity																																																								
ID	Second level cs item	CS Name	CS Description																																																					
1	1	CSItem1 Level 3	CSItem1 Level 3 Desc																																																					
2	2	CSItem2 Level 3	CSItem2 Level 3 Desc																																																					
...																																																					

Table 4.8: Pattern for re-engineering a classification scheme, which follows the snowflake data model, into an ontology(continued)

Slot	Value																																								
Example	<table border="1" style="margin-bottom: 10px;"> <thead> <tr> <th colspan="2" style="background-color: #d3d3d3;">PROFESSIONE 0</th> </tr> <tr> <th style="background-color: #d3d3d3;">ID 0</th> <th style="background-color: #d3d3d3;">Desc 0</th> </tr> </thead> <tbody> <tr> <td>01</td> <td>Professioni specialistiche e tecniche</td> </tr> <tr> <td>02</td> <td>Professioni operative della gestione d'impresa</td> </tr> <tr> <td>...</td> <td>...</td> </tr> </tbody> </table> <table border="1" style="margin-bottom: 10px;"> <thead> <tr> <th colspan="3" style="background-color: #d3d3d3;">PROFESSIONE 1</th> </tr> <tr> <th style="background-color: #d3d3d3;">ID 1</th> <th style="background-color: #d3d3d3;">Desc 1</th> <th style="background-color: #d3d3d3;">ID 0</th> </tr> </thead> <tbody> <tr> <td>01.05</td> <td>Specialist e tecnici delle scienze informatiche</td> <td>01</td> </tr> <tr> <td>02.05</td> <td>Specialist e tecnici della gestione d'impresa</td> <td>02</td> </tr> <tr> <td>...</td> <td>...</td> <td></td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th colspan="3" style="background-color: #d3d3d3;">PROFESSIONE 2</th> </tr> <tr> <th style="background-color: #d3d3d3;">ID 2</th> <th style="background-color: #d3d3d3;">Desc 2</th> <th style="background-color: #d3d3d3;">ID 1</th> </tr> </thead> <tbody> <tr> <td>01.05.01</td> <td>Specialist e tecnici delle scienze informatiche</td> <td>01</td> </tr> <tr> <td>02.05.01</td> <td>Specialist e tecnici della gestione d'impresa</td> <td>02</td> </tr> <tr> <td>...</td> <td>...</td> <td></td> </tr> </tbody> </table>	PROFESSIONE 0		ID 0	Desc 0	01	Professioni specialistiche e tecniche	02	Professioni operative della gestione d'impresa	PROFESSIONE 1			ID 1	Desc 1	ID 0	01.05	Specialist e tecnici delle scienze informatiche	01	02.05	Specialist e tecnici della gestione d'impresa	02		PROFESSIONE 2			ID 2	Desc 2	ID 1	01.05.01	Specialist e tecnici delle scienze informatiche	01	02.05.01	Specialist e tecnici della gestione d'impresa	02	
PROFESSIONE 0																																									
ID 0	Desc 0																																								
01	Professioni specialistiche e tecniche																																								
02	Professioni operative della gestione d'impresa																																								
...	...																																								
PROFESSIONE 1																																									
ID 1	Desc 1	ID 0																																							
01.05	Specialist e tecnici delle scienze informatiche	01																																							
02.05	Specialist e tecnici della gestione d'impresa	02																																							
...	...																																								
PROFESSIONE 2																																									
ID 2	Desc 2	ID 1																																							
01.05.01	Specialist e tecnici delle scienze informatiche	01																																							
02.05.01	Specialist e tecnici della gestione d'impresa	02																																							
...	...																																								
OUTPUT: Designed Ontology																																									
General	<p>The ontology generated will be based on the taxonomy architectural pattern (AP-TX-01) [SFBG⁺07]. The classification scheme item will be transformed to a class. The classification scheme item relationship will be transformed either to a <i>subClassOf</i> or a <i>partOf</i> relation. The pattern takes advantage of the use of logical patterns for asserting the <i>partOf</i> or <i>subClassOf</i> relation. Finally, the content of the classification scheme will be transformed into ontology instances.</p>																																								
Graphical Representation																																									
(UML) General Solution Ontology																																									
(UML) Example Solution Ontology																																									
PROCESS: How to Re-engineer																																									
General	<ol style="list-style-type: none"> 1. Create a class for the <i>classification scheme item</i> schema component. 2. Assert that the <i>classification scheme item</i> class is <i>subClassOf</i> itself, by using the proper logical pattern. 3. Select all the classification scheme items, by using the snowflake data model information. 4. For each one of the classification scheme items ce_i: <ol style="list-style-type: none"> 4.1. Create the corresponding ontology instance ice_i. 4.2. Assert that ice_i is instance of the <i>classification scheme item</i> class. 																																								

Table 4.8: Pattern for re-engineering a classification scheme, which follows the snowflake data model, into an ontology(continued)

Slot	Value
Example	<ol style="list-style-type: none"> 1. Create a <code>Occupation</code> class for the <i>classification scheme item</i> schema component. 2. Assert that the <code>Occupation</code> class is <i>subClassOf</i> itself, by using the proper logical pattern. <ol style="list-style-type: none"> 2.1. Create the <code>Professioni specialistiche e tecniche</code> instance and assert that it is instance of the <code>Occupation</code> class. 2.2. Create the <code>Specialist e tecnici delle scienze informatiche</code> instance and assert that it is instance of the <code>Occupation</code> class. 2.3. Create the <code>Specialist e tecnici delle gestione dimpresa</code> instance and assert that it is instance of the <code>Occupation</code> class. 2.4. Create the <code>Professioni operative della gestione dimpresa</code> instance and assert that it is instance of the <code>Occupation</code> class.
Formal Transformation	
General	Classification Scheme: $CS = \langle CSS, CSD \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $CSS \rightarrow OS$ $CSD \rightarrow KB$
Relationships	
Relations to other modelling components	Use the Architectural Pattern: TX-AP-01 [SFBG ⁺ 07]

PR-NOR-CLLO-13. Pattern for re-engineering a classification scheme, which follows the flattened data model, into an ontology. The pattern for re-engineering non-ontological resource shown in Table 4.9 suggests a guide to transform a classification scheme, which follows the flattened data model, into an ontology. The pattern transforms the resource schema into an ontology schema, and the resource content, into ontology instances.

Table 4.9: Pattern for re-engineering a classification scheme, which follows the flattened data model, into an ontology.

Slot	Value
General Information	
Name	Pattern for Re-engineering a Classification Scheme, which follows the Flattened Data Model, into an Ontology
Identifier	PR-NOR-CLLO-13
Type of Component	Pattern for Re-engineering Non-Ontological Resource (PR-NOR)
Use Case	
General	Re-engineering a classification scheme, which follows the flattened model, to design an ontology
Example	Suppose that someone wants to build an ontology based on a classification published as one table with a column for each classification level.
Pattern for Re-engineering Non-Ontological Resource	
INPUT: Resource to be Re-engineered	

Table 4.9: Pattern for re-engineering a classification scheme, which follows the flattened data model, into an ontology(continued)

Slot	Value																																										
General	<p>A non-ontological resource holds a classification scheme which follows the flattened data model.</p> <p>A classification scheme is a rooted tree of concepts, in which each concept groups entities by some particular degree of similarity. The semantics of the hierarchical relation between parents and children concepts may vary depending of the context.</p> <p>The flattened data model [MZ06] is a denormalized structure for hierarchy representations. In this case, each hierarchy level is represented on a different column. There are as many columns as levels the classification scheme has. Therefore each row has the complete path from the root to a leaf node.</p>																																										
Example	<p>The Classification of Italian Education Titles published by the National Institute of Statistics (ISTAT) is represented following a flattened model. The first level of the classification (level code) is related to the education title level which comprises values as elementary, media, university, master, etc. The second level of the classification is the type of school or institute which offers the education title. The last level is the education title itself; it has a specific specialization code and also a code which is the concatenation of the previous code levels.</p>																																										
Graphical Representation																																											
General	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="7" style="background-color: #333; color: white;">Flattened Entity</th> </tr> <tr> <th colspan="2" style="background-color: #333; color: white;">First level</th> <th colspan="2" style="background-color: #333; color: white;">Second level</th> <th colspan="2" style="background-color: #333; color: white;">Third level</th> <th style="background-color: #333; color: white;">...</th> </tr> <tr> <th style="background-color: #333; color: white;">ID</th> <th style="background-color: #333; color: white;">CS Name</th> <th style="background-color: #333; color: white;">ID</th> <th style="background-color: #333; color: white;">CS Name</th> <th style="background-color: #333; color: white;">ID</th> <th style="background-color: #333; color: white;">CS Name</th> <th style="background-color: #333; color: white;">...</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>CSItem1 Level 1</td> <td>1</td> <td>CSItem1 Level 2</td> <td>1</td> <td>CSItem1 Level 3</td> <td>...</td> </tr> <tr> <td>2</td> <td>CSItem2 Level 1</td> <td>2</td> <td>CSItem2 Level 2</td> <td>2</td> <td>CSItem2 Level 3</td> <td>...</td> </tr> <tr> <td>...</td> <td>...</td> <td>...</td> <td>...</td> <td>...</td> <td>...</td> <td>..</td> </tr> </tbody> </table>	Flattened Entity							First level		Second level		Third level		...	ID	CS Name	ID	CS Name	ID	CS Name	...	1	CSItem1 Level 1	1	CSItem1 Level 2	1	CSItem1 Level 3	...	2	CSItem2 Level 1	2	CSItem2 Level 2	2	CSItem2 Level 3
Flattened Entity																																											
First level		Second level		Third level		...																																					
ID	CS Name	ID	CS Name	ID	CS Name	...																																					
1	CSItem1 Level 1	1	CSItem1 Level 2	1	CSItem1 Level 3	...																																					
2	CSItem2 Level 1	2	CSItem2 Level 2	2	CSItem2 Level 3	...																																					
...																																					
Example	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="7" style="background-color: #333; color: white;">ISTAT - CTS103 - Italian Education Titles Classification</th> </tr> <tr> <th style="background-color: #333; color: white;">ID Education</th> <th style="background-color: #333; color: white;">Code of school or academic course specialization</th> <th style="background-color: #333; color: white;">Education Title</th> <th style="background-color: #333; color: white;">Code of school, institute or group.</th> <th style="background-color: #333; color: white;">Type of school, institute or group</th> <th style="background-color: #333; color: white;">Level ID</th> <th style="background-color: #333; color: white;">...</th> </tr> </thead> <tbody> <tr> <td>30101010</td> <td>010</td> <td>Esperto frutticoltore</td> <td>101</td> <td>Istituto professionale agrario</td> <td>30</td> <td></td> </tr> <tr> <td>30101011</td> <td>011</td> <td>Esperto olivicoltore</td> <td>101</td> <td>Istituto professionale agrario</td> <td>30</td> <td></td> </tr> <tr> <td>40104001</td> <td>001</td> <td>Analista contabile</td> <td>104</td> <td>Istit. Profess.Servizi Commerciali</td> <td>40</td> <td></td> </tr> <tr> <td>40104002</td> <td>002</td> <td>Operatore commerciale</td> <td>104</td> <td>Istit. Profess.Servizi Commerciali</td> <td>40</td> <td></td> </tr> </tbody> </table>	ISTAT - CTS103 - Italian Education Titles Classification							ID Education	Code of school or academic course specialization	Education Title	Code of school, institute or group.	Type of school, institute or group	Level ID	...	30101010	010	Esperto frutticoltore	101	Istituto professionale agrario	30		30101011	011	Esperto olivicoltore	101	Istituto professionale agrario	30		40104001	001	Analista contabile	104	Istit. Profess.Servizi Commerciali	40		40104002	002	Operatore commerciale	104	Istit. Profess.Servizi Commerciali	40	
ISTAT - CTS103 - Italian Education Titles Classification																																											
ID Education	Code of school or academic course specialization	Education Title	Code of school, institute or group.	Type of school, institute or group	Level ID	...																																					
30101010	010	Esperto frutticoltore	101	Istituto professionale agrario	30																																						
30101011	011	Esperto olivicoltore	101	Istituto professionale agrario	30																																						
40104001	001	Analista contabile	104	Istit. Profess.Servizi Commerciali	40																																						
40104002	002	Operatore commerciale	104	Istit. Profess.Servizi Commerciali	40																																						
OUTPUT: Designed Ontology																																											
General	<p>The ontology generated will be based on the taxonomy architectural pattern (AP-TX-01) [SFBG⁺07]. The classification scheme item will be transformed to a class. The classification scheme item relationship will be transformed either to a <i>subClassOf</i> or a <i>partOf</i> relation. The pattern takes advantage of the use of logical patterns for asserting the <i>partOf</i> or <i>subClassOf</i> relation. Finally, the content of the classification scheme will be transformed into ontology instances.</p>																																										
Graphical Representation																																											
(UML) General Solution Ontology	<pre> classDiagram class Class0 class Instance1 class Instance2 class Instance3 Class0 --> Class0 Instance1 -- > Class0 : type Instance2 -- > Class0 : type Instance3 -- > Class0 : type </pre>																																										

Table 4.9: Pattern for re-engineering a classification scheme, which follows the flattened data model, into an ontology(continued)

Slot	Value
(UML) Example Solution Ontology	<pre> classDiagram class EducationTitles class HigherSecondaryEducation["HIGHER SECONDARY EDUCATION"] class HigherSecondaryEducationAllowsAccess["HIGHER SECONDARY EDUCATION - ALLOWS ACCESS TO UNIVERSITIES"] class IstitutoProfessionaleAgrario["Istituto professionale agrario"] class EspertoFrutticoltore["Esperto frutticoltore"] class EspertoOlivicoltore["Esperto olivicoltore"] EducationTitles < -- HigherSecondaryEducation EducationTitles < -- HigherSecondaryEducationAllowsAccess EducationTitles < -- IstitutoProfessionaleAgrario EducationTitles < -- EspertoFrutticoltore EducationTitles < -- EspertoOlivicoltore HigherSecondaryEducation -- > HigherSecondaryEducationAllowsAccess : type </pre>
PROCESS: How to Re-engineer	
General	<ol style="list-style-type: none"> 1. Create a class for the <i>classification scheme item</i> schema component. 2. Assert that the <i>classification scheme item</i> class is <i>subClassOf</i> itself, by using the proper logical pattern. 3. Select all the classification scheme items, by using the flattened data model information. 4. For each one of the classification scheme items ce_i: <ol style="list-style-type: none"> 4.1. Create the corresponding ontology instance ice_i. 4.2. Assert that ice_i is instance of the <i>classification scheme item</i> class.
Example	<ol style="list-style-type: none"> 1. Create a <code>Education Title</code> class for the <i>classification scheme item</i> schema component. 2. Assert that the <code>Education Title</code> class is <i>subClassOf</i> itself, by using the proper logical pattern. <ol style="list-style-type: none"> 2.1. Create the <code>HIGHER SECONDARY EDUCATION</code> instance and assert that it is instance of the <code>Education Title</code> class. 2.2. Create the <code>Istituto professionale agrario</code> instance and assert that it is instance of the <code>Education Title</code> class. 2.3. Create the <code>Esperto frutticoltore</code> instance and assert that it is instance of the <code>Education Title</code> class. 2.4. Create the <code>Esperto olivicoltore</code> instance and assert that it is instance of the <code>Education Title</code> class. 2.5. Create the <code>HIGHER SECONDARY EDUCATION - ALLOWS ACCESS TO UNIVERSITIES</code> instance and assert that it is instance of the <code>Education Title</code> class. 2.6. Create the <code>Education Title</code> instance and assert that it is instance of the <code>Education Title</code> class.
Formal Transformation	
General	<p>Classification Scheme: $CS = \langle CSS, CSD \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $CSS \rightarrow OS$ $CSD \rightarrow KB$</p>
Relationships	
Relations to other modelling components	<p>Use the Architectural Pattern: TX-AP-01 [SFBG⁺07]</p>

4.4.2 Patterns for Re-engineering Thesauri into Ontologies

In this section we present the re-engineering patterns (PR-NOR) for re-engineering thesauri into ontologies. The patterns are:

- Patterns for the TBox transformations
 - PR-NOR-TSLO-01. Pattern for re-engineering a term-based thesaurus, which follows the record-based data model, into an ontology schema.
 - PR-NOR-TSLO-02. Pattern for re-engineering a term-based thesaurus, which follows the relation-based data model, into an ontology schema.
- Patterns for the ABox transformations
 - PR-NOR-TSLO-11. Pattern for re-engineering a term-based thesaurus, which follows the record-based data model, into an ontology.
 - PR-NOR-TSLO-12. Pattern for re-engineering a term-based thesaurus, which follows the relation-based data model, into an ontology.

Patterns for the TBox transformations

These patterns transform the resource content into an ontology schema. The TBox transformation approach tries to enforce a formal semantics to the re-engineered resources, even at the cost of changing their structure [SAd⁺07a]. For the disambiguation of the semantics of the BT, NT and RT relations among thesaurus terms the patterns rely on an external resource, e.g. Scarlet and WordNet. In the case of that the external resource does not provide any relation between two terms, the patterns take advantage of the use of logical patterns⁹ for asserting the relation *partOf*, *subClassOf* or *relatedClass*. For the UF/USE relations we use the logical pattern proposed by Corcho et al. [CR09] suggested as best practice in the context of this antipattern: the tendency to declare two classes equivalent when in fact their labels simply express synonymy.

PR-NOR-TSLO-01. Pattern for re-engineering a term-based thesaurus, which follows the record-based data model, into an ontology schema. The pattern for re-engineering thesaurus shown in Table 4.10 suggests a guide to transform a thesaurus into an ontology schema. The thesaurus is a term-based one and it is modeled with a record-based data model.

Table 4.10: Pattern for re-engineering a term-based thesaurus, which follows the record-based data model, into an ontology schema

Slot	Value
General Information	
Name	Pattern for Re-engineering a Term-based Thesaurus, which follows the Record-based Data Model, into an Ontology Schema.
Identifier	PR-NOR-TSLO-01
Type of Component	Pattern for Re-engineering Non-Ontological Resources (PR-NOR)
Use Case	
General	Re-engineering a term-based thesaurus which follows the record-based model to design an ontology schema.
Example	Suppose that someone wants to build an ontology schema based on the European Training Thesaurus (ETT), which is a term-based thesaurus and it follows the record-based model.
Pattern for Re-engineering Non-Ontological Resources	

⁹Logical patterns are included in the ODP portal at <http://ontologydesignpatterns.org>

Table 4.10: Pattern for re-engineering a term-based thesaurus, which follows the record-based data model, into an ontology schema (continued)

Slot	Value															
INPUT: Resource to be Re-engineered																
General	<p>A non-ontological resource holds a term-based thesaurus which follows the record-based model.</p> <p>A thesaurus represents the knowledge of a domain with a collection of terms and a limited set of relations between them.</p> <p>The record-based data model [Soe95] is a denormalized structure, uses a record for every term with the information about the term, such as synonyms, broader, narrower and related terms.</p>															
Example	<p>The European Training Thesaurus (ETT) constitutes the controlled vocabulary of reference in the field of vocational education and training (VET) in Europe.</p> <p>This thesaurus is available at http://libserver.cedefop.europa.eu/ett/en/.</p>															
Graphical Representation																
General	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="background-color: #cccccc;">Term</th> <th style="background-color: #cccccc;">BT</th> <th style="background-color: #cccccc;">NT</th> <th style="background-color: #cccccc;">RT</th> <th style="background-color: #cccccc;">UF</th> </tr> </thead> <tbody> <tr> <td>Term1</td> <td>BTterm1</td> <td>NTTerm1 NTTerm2</td> <td>Term2</td> <td>UFTerm1</td> </tr> <tr> <td>Term2</td> <td>BTterm2</td> <td>NTTerm3 NTTerm4 NTTerm5 NTTerm6 NTTerm7 NTTerm8 NTTerm9 NTTerm10</td> <td>RTTerm3 RTTerm4 RTTerm5</td> <td></td> </tr> </tbody> </table>	Term	BT	NT	RT	UF	Term1	BTterm1	NTTerm1 NTTerm2	Term2	UFTerm1	Term2	BTterm2	NTTerm3 NTTerm4 NTTerm5 NTTerm6 NTTerm7 NTTerm8 NTTerm9 NTTerm10	RTTerm3 RTTerm4 RTTerm5	
Term	BT	NT	RT	UF												
Term1	BTterm1	NTTerm1 NTTerm2	Term2	UFTerm1												
Term2	BTterm2	NTTerm3 NTTerm4 NTTerm5 NTTerm6 NTTerm7 NTTerm8 NTTerm9 NTTerm10	RTTerm3 RTTerm4 RTTerm5													
Example	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="background-color: #cccccc;">Term</th> <th style="background-color: #cccccc;">BT</th> <th style="background-color: #cccccc;">NT</th> <th style="background-color: #cccccc;">RT</th> <th style="background-color: #cccccc;">UF</th> </tr> </thead> <tbody> <tr> <td>competence</td> <td>learning</td> <td>skill</td> <td>aptitude know how knowledge performance</td> <td></td> </tr> <tr> <td>performance</td> <td>personal development</td> <td>efficiency failure success</td> <td>competence productivity</td> <td>achievement</td> </tr> </tbody> </table>	Term	BT	NT	RT	UF	competence	learning	skill	aptitude know how knowledge performance		performance	personal development	efficiency failure success	competence productivity	achievement
Term	BT	NT	RT	UF												
competence	learning	skill	aptitude know how knowledge performance													
performance	personal development	efficiency failure success	competence productivity	achievement												
OUTPUT: Designed Ontology																
General	<p>The generated ontology will be based on the lightweight ontology architectural pattern (AP-LW-01)[SFBG⁺07]. Each thesaurus term is mapped to a class. For the disambiguation of the semantics of the BT, NT and RT relations among thesaurus terms the pattern relies on an external resource. In the case of that the external resource does not provide any relation between two terms, the pattern takes advantage of the use of logical patterns for asserting the relation <i>partOf</i>, <i>subclassOf</i> or <i>relatedClass</i>. For the UF/USE relations we use the logical pattern proposed by Corcho et al. [CR09] suggested as best practice in the context of this antipattern: the tendency to declare two classes equivalent when in fact their labels simply express synonym.</p>															
Graphical Representation																
(UML) General Solution Ontology	<pre> classDiagram class Term1 { -rdfs:label UFTerm1 } class Term2 class BTTerm1 class BTTerm2 Term1 < -- BTTerm1 Term1 < -- BTTerm2 Term1 --> Term2 : relatedClass Note over Term1, Term2: <<rdfs:domain>> Note over Term1, Term2: <<rdfs:range>> Note over Term1, Term2: <<owl:ObjectProperty>> relatedClass </pre>															

Table 4.10: Pattern for re-engineering a term-based thesaurus, which follows the record-based data model, into an ontology schema (continued)

Slot	Value
<p>(UML) Example Solution Ontology</p>	<pre> classDiagram class learning class personal_development class competence class performance class skill class efficiency class failure class success learning < -- competence personal_development < -- performance competence < -- skill competence <--> performance : relatedClass performance < -- efficiency performance < -- failure performance < -- success performance <.. achievement : rdfs:label </pre>
<p>PROCESS: How to Re-engineer</p>	
<p>General</p>	<ol style="list-style-type: none"> 1. Identify the records which contain thesaurus terms without a <i>broader term</i>. 2. For each one of the above identified thesaurus terms t_i: <ol style="list-style-type: none"> 2.1. Create the corresponding ontology class, C_i class, if it is not created yet. 2.2. Identify the thesaurus terms, t_j, which are narrower terms of t_i. They are referenced in the same record which contains t_i. 2.3. For each one of the above identified thesaurus term t_j: <ol style="list-style-type: none"> 2.3.1. Create the corresponding ontology class, C_j class, if it is not created yet. 2.3.2. Using the external resource identify the semantics of the relation between C_j and C_i, and set up the relation identified. 2.3.3. Repeat from step 2.2 for c_j as a new c_i 2.4. Identify the thesaurus terms, t_r, which are related terms of t_i. They are referenced in the same record which contains t_i. 2.5. For each one of the above identified thesaurus terms t_r: <ol style="list-style-type: none"> 2.5.1. Create the corresponding ontology class, C_r class, if it is not created yet. 2.5.2. Using the external resource identify the semantics of the relation between C_r and C_i, and set up the relation identified. 2.5.3. Repeat from step 2.4 for c_r as a new c_i 2.6. Identify the thesaurus terms, t_q, which are equivalent terms of t_i. They are referenced in the same record which contains t_i. 2.7. For each one of the above identified thesaurus terms t_q: <ol style="list-style-type: none"> 2.7.1. Use the logical pattern proposed by Corcho et al. [CR09]. 2.7.2. Repeat from step 2.6 for c_q as a new c_i.

Table 4.10: Pattern for re-engineering a term-based thesaurus, which follows the record-based data model, into an ontology schema (continued)

Slot	Value
Example	<ol style="list-style-type: none"> 1. Create the <code>learning</code> class and the <code>personal development</code> class. 2. Create the <code>competence</code> class. 3. Using the external resource identify the semantics of the relation between <code>competence</code> and <code>learning</code>, and set up the relation identified. 4. Create the <code>performance</code> class. 5. Using the external resource identify the semantics of the relation between <code>performance</code> and <code>personal development</code>, and set up the relation identified. 6. Assert that <code>achievement</code> is label of the <code>performance</code> class. 7. Using the external resource identify the semantics of the relation between <code>competence</code> and <code>performance</code>, and set up the relation identified. 8. Create the <code>skill</code> class. 9. Using the external resource identify the semantics of the relation between <code>skill</code> and <code>competence</code>, and set up the relation identified. <ol style="list-style-type: none"> 9.1. Create the <code>efficiency</code> class. 9.2. Using the external resource identify the semantics of the relation between <code>efficiency</code> and <code>performance</code>, and set up the relation identified. 9.3. Create the <code>failure</code> class. 9.4. Using the external resource identify the semantics of the relation between <code>failure</code> and <code>performance</code>, and set up the relation identified. 9.5. Create the <code>success</code> class. 9.6. Using the external resource identify the semantics of the relation between <code>success</code> and <code>performance</code>, and set up the relation identified.
Formal Transformation	
General	Thesaurus: $T = \langle TS, TD \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $TD \rightarrow OS$
Relationships	
Relations to other modelling components	Use the Architectural Pattern: AP-LW-01 [SFBG ⁺ 07]

PR-NOR-TSLO-02. Pattern for re-engineering a term-based thesaurus, which follows the relation-based data model, into an ontology schema The pattern for re-engineering thesaurus shown in Table 4.11 suggests a guide to transform a thesaurus into an ontology schema. The thesaurus is a term-based one and it is modeled with a relation-based data model.

Table 4.11: Pattern for re-engineering a term-based thesaurus, which follows the relation-based model, into an ontology schema

Slot	Value
General Information	
Name	Pattern for Re-engineering a Term-based Thesaurus, which follows the Relation-based Model, into an Ontology Schema
Identifier	PR-NOR-TSLO-02
Type of Component	Pattern for Re-engineering Non-Ontological Resources (PR-NOR)
Use Case	
General	Re-engineering a term-based thesaurus, which follows the relation-based model, to design an ontology schema.

Table 4.11: Pattern for re-engineering a term-based thesaurus, which follows the relation-based model, into an ontology schema (continued)

Slot	Value																					
Example	Suppose that someone wants to build an ontology schema based on earlier version of the AGROVOC Thesaurus, which is a term-based thesaurus and it follows the relation-based model.																					
Pattern for Re-engineering Non-Ontological Resources																						
INPUT: Resource to be Re-engineered																						
General	A non-ontological resource holds a term-based thesaurus which follows the relation-based model. A thesaurus represents the knowledge of a domain with a collection of terms and a limited set of relations between them. The relation-based data model [Soe95] is a normalized structure, in which relationship types are not defined as fields in a record, but they are simply data values in a relationship record, thus new relationship types can be introduced with ease.																					
Example	The AGROVOC Thesaurus is an structured and controlled vocabulary designed to cover the terminology of all subject fields in agriculture, forestry, fisheries, food and related domains. This thesaurus is available at http://www.fao.org/agrovoc/ .																					
Graphical Representation																						
General	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="background-color: black; color: white;">(1) Term Entity</th> <th style="background-color: black; color: white;">(2) Term-Term Relationship Entity</th> <th style="background-color: black; color: white;">(3) Relationship Entity</th> </tr> <tr> <th style="background-color: black; color: white;">TermCode Term</th> <th style="background-color: black; color: white;">TermCode1 TermCode2 ReIID</th> <th style="background-color: black; color: white;">ReIID RelDesc RelAbr</th> </tr> </thead> <tbody> <tr> <td>1001 Term1</td> <td>1001 1003 10</td> <td>10 Broader Term BT</td> </tr> <tr> <td>1002 Term2</td> <td>1003 1004 20</td> <td>30 Related Term RT</td> </tr> <tr> <td>1003 Term3</td> <td>1002 1005 10</td> <td>20 Used For UF</td> </tr> <tr> <td>1004 Term4</td> <td>1003 1005 30</td> <td></td> </tr> <tr> <td>1005 Term5</td> <td></td> <td></td> </tr> </tbody> </table>	(1) Term Entity	(2) Term-Term Relationship Entity	(3) Relationship Entity	TermCode Term	TermCode1 TermCode2 ReIID	ReIID RelDesc RelAbr	1001 Term1	1001 1003 10	10 Broader Term BT	1002 Term2	1003 1004 20	30 Related Term RT	1003 Term3	1002 1005 10	20 Used For UF	1004 Term4	1003 1005 30		1005 Term5		
(1) Term Entity	(2) Term-Term Relationship Entity	(3) Relationship Entity																				
TermCode Term	TermCode1 TermCode2 ReIID	ReIID RelDesc RelAbr																				
1001 Term1	1001 1003 10	10 Broader Term BT																				
1002 Term2	1003 1004 20	30 Related Term RT																				
1003 Term3	1002 1005 10	20 Used For UF																				
1004 Term4	1003 1005 30																					
1005 Term5																						
Example	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="background-color: black; color: white;">(1) agrovocterm</th> <th style="background-color: black; color: white;">(2) termlink</th> <th style="background-color: black; color: white;">(3) linktype</th> </tr> <tr> <th style="background-color: black; color: white;">TermCode Term</th> <th style="background-color: black; color: white;">TermCode1 TermCode2 LinktypeID</th> <th style="background-color: black; color: white;">LinktypeID LinkDesc LinkAbr</th> </tr> </thead> <tbody> <tr> <td>1328 Paddy</td> <td>5435 6599 90</td> <td>50 Broader Term BT</td> </tr> <tr> <td>1474 Cereals</td> <td>5435 3354 50</td> <td>90 Related Term RT</td> </tr> <tr> <td>3354 Poaceae</td> <td>6599 5435 90</td> <td>60 Narrower Term NT</td> </tr> <tr> <td>5435 Oryza</td> <td>6599 1474 50</td> <td>20 Used For UF</td> </tr> <tr> <td>6599 Rice</td> <td>6599 1328 20</td> <td></td> </tr> </tbody> </table>	(1) agrovocterm	(2) termlink	(3) linktype	TermCode Term	TermCode1 TermCode2 LinktypeID	LinktypeID LinkDesc LinkAbr	1328 Paddy	5435 6599 90	50 Broader Term BT	1474 Cereals	5435 3354 50	90 Related Term RT	3354 Poaceae	6599 5435 90	60 Narrower Term NT	5435 Oryza	6599 1474 50	20 Used For UF	6599 Rice	6599 1328 20	
(1) agrovocterm	(2) termlink	(3) linktype																				
TermCode Term	TermCode1 TermCode2 LinktypeID	LinktypeID LinkDesc LinkAbr																				
1328 Paddy	5435 6599 90	50 Broader Term BT																				
1474 Cereals	5435 3354 50	90 Related Term RT																				
3354 Poaceae	6599 5435 90	60 Narrower Term NT																				
5435 Oryza	6599 1474 50	20 Used For UF																				
6599 Rice	6599 1328 20																					
OUTPUT: Designed Ontology																						
General	The generated ontology will be based on the lightweight ontology architectural pattern (AP-LW-01)[SFBG ⁺ 07]. Each thesaurus term is mapped to a class. For the disambiguation of the semantics of the BT, NT and RT relations among thesaurus terms the pattern relies on an external resource. In the case the external resource does not provide any relation between two terms, the pattern takes advantage of the use of logical patterns for asserting the relation <i>partOf</i> , <i>subClassOf</i> or <i>relatedClass</i> . For the UF/USE relations we use the logical pattern proposed by Corcho et al. [CR09] suggested as best practice in the context of this antipattern: the tendency to declare two classes equivalent when in fact their labels simply express synonym.																					
Graphical Representation																						
(UML) General Solution Ontology	<pre> classDiagram class Term1 class Term2 class Term3 class Term5 Term1 -- > Term3 Term2 -- > Term5 Term3 -- Term5 : relatedClass Term3 ..> Term5 : <<rdfs:domain>> Term5 ..> Term3 : <<rdfs:range>> </pre>																					

Table 4.11: Pattern for re-engineering a term-based thesaurus, which follows the relation-based model, into an ontology schema (continued)

Slot	Value
<p>(UML) Example Solution Ontology</p>	
<p>PROCESS: How to Re-engineer</p>	
<p>General</p>	<ol style="list-style-type: none"> 1. Identify the records which contain thesaurus terms without a <i>broader term</i>, within the term-term relationship entity. 2. For each one of the above identified thesaurus terms t_i: <ol style="list-style-type: none"> 2.1. Obtain the thesaurus term within the term entity. 2.2. Create the corresponding ontology class, C_i class, if it is not created yet. 2.3. Identify the thesaurus term, t_j, which are narrower terms of t_i, within the term-term relationship entity. 2.4. For each one of the above identified thesaurus terms t_j: <ol style="list-style-type: none"> 2.4.1. Obtain the thesaurus term within the term entity. 2.4.2. Create the corresponding ontology class, C_j class, if it is not created yet. 2.4.3. Using the external resource identify the semantics of the relation between C_j and C_i, and set up the relation identified. 2.4.4. Repeat from step 2.2 for c_j as a new c_i 2.5. Identify the thesaurus term, t_r, which are related terms of t_i, within the term-term relationship entity. 2.6. For each one of the above identified thesaurus term t_r: <ol style="list-style-type: none"> 2.6.1. Obtain the thesaurus term within the term entity. 2.6.2. Create the corresponding ontology class, C_r class, if it is not created yet. 2.6.3. Using the external resource identify the semantics of the relation between C_r and C_i, and set up the relation identified. 2.6.4. Repeat from step 2.4 for c_r as a new c_i 2.7. Identify the thesaurus term, t_q, which are equivalent terms of t_i, within the term-term relationship entity. 2.8. For each one of the above identified thesaurus term t_q: <ol style="list-style-type: none"> 2.8.1. Use the logical pattern proposed by Corcho et al. [CR09]. 2.8.2. Repeat from step 2.6 for c_q as a new c_i
<p>Example</p>	<ol style="list-style-type: none"> 1. Create the <code>Poaceae</code> class. <ol style="list-style-type: none"> 1.1. Create the <code>Oryza</code> class. 1.2. Using the external resource identify the semantics of the relation between <code>Oryza</code> and <code>Poaceae</code>, and set up the relation identified. <ol style="list-style-type: none"> 1.2.1. Create the <code>Rice</code> class. 1.2.2. Using the external resource identify the semantics of the relation between <code>Rice</code> and <code>Oryza</code>, and set up the relation identified. 2. Create the <code>Cereals</code> class. <ol style="list-style-type: none"> 2.1. Using the external resource identify the semantics of the relation between <code>Rice</code> and <code>Cereals</code>, and set up the relation identified. 2.2. Assert that <code>Rice</code> is label of the <code>Paddy</code> class.
<p>Formal Transformation</p>	

Table 4.11: Pattern for re-engineering a term-based thesaurus, which follows the relation-based model, into an ontology schema (continued)

Slot	Value
General	Thesaurus: $T = \langle TS, TD \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $TD \rightarrow OS$
Relationships	
Relations to other modelling components	Use the Architectural Pattern: AP-LW-01 [SFBG ⁺ 07]

Patterns for the ABox transformations

These patterns transform the resource schema into an ontology schema, and the resource content, into ontology instances. The ABox transformation approach leaves the informal semantics of the re-engineered resources mostly untouched [SAd⁺07a].

These patterns deal with term-based thesauri. According to [VTAGS⁺08], the schema of a term-based thesaurus has the following main components: (1) PreferredTerm, which will be transformed to a class, (2) Hierarchical Relationship, which will be transformed either to a *subClassOf* or *partOf* relation, (3) Associative Relationship, which will be transformed to an *ad-hoc* relation, (4) Equivalent terms, the ones from the USE relationships, which will be transformed to labels, by using the logical pattern proposed by Corcho et al. [CR09]. The patterns take advantage of the use of logical patterns¹⁰ for asserting the *partOf* or *subClassOf* relation. Finally, the content of the thesaurus will be transformed into ontology instances.

PR-NOR-TSLO-10. Pattern for re-engineering a term-based thesaurus, which follows the record-based data model, into an ontology. The pattern for re-engineering thesaurus shown in Table 4.12 suggests a guide to transform a term-based thesaurus, which follows the record-based data model, into an ontology. The pattern transforms the resource schema into an ontology schema, and the resource content, into ontology instances.

Table 4.12: Pattern for re-engineering a term-based thesaurus, which follows the record-based data model, into an ontology

Slot	Value
General Information	
Name	Pattern for Re-engineering a Term-based Thesaurus, which follows the Record-based Data Model, into an Ontology.
Identifier	PR-NOR-TSLO-10
Type of Component	Pattern for Re-engineering Non-Ontological Resources (PR-NOR)
Use Case	
General	Re-engineering a term-based thesaurus which follows the record-based model to design an ontology.
Example	Suppose that someone wants to build an ontology based on the European Training Thesaurus (ETT), which is a term-based thesaurus and it follows the record-based model.
Pattern for Re-engineering Non-Ontological Resources	
INPUT: Resource to be Re-engineered	

¹⁰Logical patterns are included in the ODP portal at <http://ontologydesignpatterns.org>

Table 4.12: Pattern for re-engineering a term-based thesaurus, which follows the record-based data model, into an ontology (continued)

Slot	Value															
General	<p>A non-ontological resource holds a term-based thesaurus which follows the record-based model.</p> <p>A thesaurus represents the knowledge of a domain with a collection of terms and a limited set of relations between them.</p> <p>The record-based data model [Soe95] is a denormalized structure, uses a record for every term with the information about the term, such as synonyms, broader, narrower and related terms.</p>															
Example	<p>The European Training Thesaurus (ETT) constitutes the controlled vocabulary of reference in the field of vocational education and training (VET) in Europe.</p> <p>This thesaurus is available at http://libserver.cedefop.europa.eu/ett/en/.</p>															
Graphical Representation																
General	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="background-color: #cccccc;">Term</th> <th style="background-color: #cccccc;">BT</th> <th style="background-color: #cccccc;">NT</th> <th style="background-color: #cccccc;">RT</th> <th style="background-color: #cccccc;">UF</th> </tr> </thead> <tbody> <tr> <td>Term1</td> <td>BTterm1</td> <td>NTTerm1 NTTerm2</td> <td>Term2</td> <td>UFTerm1</td> </tr> <tr> <td>Term2</td> <td>BTterm2</td> <td>NTTerm3 NTTerm4 NTTerm5 NTTerm6 NTTerm7 NTTerm8 NTTerm9 NTTerm10</td> <td>RTTerm3 RTTerm4 RTTerm5</td> <td></td> </tr> </tbody> </table>	Term	BT	NT	RT	UF	Term1	BTterm1	NTTerm1 NTTerm2	Term2	UFTerm1	Term2	BTterm2	NTTerm3 NTTerm4 NTTerm5 NTTerm6 NTTerm7 NTTerm8 NTTerm9 NTTerm10	RTTerm3 RTTerm4 RTTerm5	
Term	BT	NT	RT	UF												
Term1	BTterm1	NTTerm1 NTTerm2	Term2	UFTerm1												
Term2	BTterm2	NTTerm3 NTTerm4 NTTerm5 NTTerm6 NTTerm7 NTTerm8 NTTerm9 NTTerm10	RTTerm3 RTTerm4 RTTerm5													
Example	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="background-color: #cccccc;">Term</th> <th style="background-color: #cccccc;">BT</th> <th style="background-color: #cccccc;">NT</th> <th style="background-color: #cccccc;">RT</th> <th style="background-color: #cccccc;">UF</th> </tr> </thead> <tbody> <tr> <td>competence</td> <td>learning</td> <td>skill</td> <td>aptitude know how knowledge performance</td> <td></td> </tr> <tr> <td>performance</td> <td>personal development</td> <td>efficiency failure success</td> <td>competence productivity</td> <td>achievement</td> </tr> </tbody> </table>	Term	BT	NT	RT	UF	competence	learning	skill	aptitude know how knowledge performance		performance	personal development	efficiency failure success	competence productivity	achievement
Term	BT	NT	RT	UF												
competence	learning	skill	aptitude know how knowledge performance													
performance	personal development	efficiency failure success	competence productivity	achievement												
OUTPUT: Designed Ontology																
General	<p>The generated ontology will be based on the lightweight ontology architectural pattern (AP-LW-01)[SFBG⁺07]. The thesaurus Term, schema component, will be transformed to a class, the hierarchical relationship will be transformed either to a <i>subClassOf</i> or <i>partOf</i> relation, the associative relationship will be transformed to an <i>ad-hoc</i> relation, and equivalent terms, the ones from the USE relationships, will be transformed to labels, by using the logical pattern proposed by Corcho et al. [CR09]. Finally, the content of the thesaurus will be transformed into ontology instances.</p>															
Graphical Representation																
(UML) General Solution Ontology	<pre> classDiagram class PreferredTerm { +self-referencing association } class Term1 { -rdfs:label UFTerm1 } class BTterm1 class Term2 class BTterm2 PreferredTerm < -- Term1 PreferredTerm < -- BTterm1 PreferredTerm < -- Term2 PreferredTerm < -- BTterm2 </pre>															

Table 4.12: Pattern for re-engineering a term-based thesaurus, which follows the record-based data model, into an ontology (continued)

Slot	Value
<p>(UML) Example Solution Ontology</p>	<pre> classDiagram class VocationalEducation["Vocational education"] class learning class success class competence class personalDevelopment["personal development"] class performance class skill class efficiency class failure VocationalEducation -- > learning VocationalEducation -- > success VocationalEducation -- > competence VocationalEducation -- > personalDevelopment learning -- > performance learning -- > skill success -- > efficiency success -- > failure performance -.-> achievement["-rdfs:label achievement"] VocationalEducation --> VocationalEducation : type </pre>
PROCESS: How to Re-engineer	
<p>General</p>	<ol style="list-style-type: none"> 1. Create a class for the <i>PreferredTerm</i> schema component. 2. Assert that the <i>PreferredTerm</i> class is <i>subClassOf</i> itself, by using the proper logical pattern. 3. Assert that the <i>PreferredTerm</i> class is <i>relatedTo</i> itself. 4. Select all the preferred terms, by using the record-based data model information. 5. For each one of the preferred terms t_i: <ol style="list-style-type: none"> 5.1. Create the corresponding ontology instance it_i. 5.2. Assert that it_i is instance of the <i>PreferredTerm</i> class. 5.3. Identify the thesaurus terms, t_q, which are equivalent terms of t_i. They are referenced in the same record which contains t_i. 5.4. For each one of the above identified thesaurus terms t_q: <ol style="list-style-type: none"> 5.4.1. Use the logical pattern proposed by Corcho et al. [CR09].
<p>Example</p>	<ol style="list-style-type: none"> 1. Create a <code>Vocational education</code> class for the <i>PreferredTerm</i> schema component. 2. Assert that the <code>Vocational education</code> class is <i>subClassOf</i> itself, by using the proper logical pattern. 3. Assert that the <code>Vocational education</code> class is <i>relatedTo</i> itself. 4. Create the <code>learning</code> instance and assert that it is instance of the <code>Vocational education</code> class. 5. Create the <code>personal development</code> instance and assert that it is instance of the <code>Vocational education</code> class. 6. Create the <code>competence</code> instance and assert that it is instance of the <code>Vocational education</code> class. 7. Create the <code>performance</code> instance and assert that it is instance of the <code>Vocational education</code> class. 8. Assert that <code>achievement</code> is label of the <code>performance</code> class. 9. Create the <code>efficiency</code> instance and assert that it is instance of the <code>Vocational education</code> class. 10. Create the <code>failure</code> instance and assert that it is instance of the <code>Vocational education</code> class. 11. Create the <code>success</code> instance and assert that it is instance of the <code>Vocational education</code> class.
Formal Transformation	
<p>General</p>	<p>Thesaurus: $T = \langle TS, TD \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $TS \rightarrow OS$ $TD \rightarrow KB$</p>

Table 4.12: Pattern for re-engineering a term-based thesaurus, which follows the record-based data model, into an ontology (continued)

Slot	Value
Relationships	
Relations to other modelling components	Use the Architectural Pattern: AP-LW-01 [SFBG ⁺ 07]

PR-NOR-TSLO-11. Pattern for re-engineering a term-based thesaurus, which follows the relation-based data model, into an ontology. The pattern for re-engineering thesaurus shown in Table 4.13 suggests a guide to transform a term-based thesaurus, which follows the record-based data model, into an ontology. The pattern transforms the resource schema into an ontology schema, and the resource content, into ontology instances.

Table 4.13: Pattern for re-engineering a term-based thesaurus, which follows the relation-based model, into an ontology

Slot	Value																																																								
General Information																																																									
Name	Pattern for Re-engineering a Term-based Thesaurus, which follows the Relation-based Model, into an Ontology																																																								
Identifier	PR-NOR-TSLO-11																																																								
Type of Component	Pattern for Re-engineering Non-Ontological Resources (PR-NOR)																																																								
Use Case																																																									
General	Re-engineering a term-based thesaurus, which follows the relation-based model, to design an ontology																																																								
Example	Suppose that someone wants to build an ontology based on earlier version of the AGROVOC Thesaurus, which is a term-based thesaurus and it follows the relation-based model.																																																								
Pattern for Re-engineering Non-Ontological Resources																																																									
INPUT: Resource to be Re-engineered																																																									
General	A non-ontological resource holds a term-based thesaurus which follows the relation-based model. A thesaurus represents the knowledge of a domain with a collection of terms and a limited set of relations between them. The relation-based data model [Soe95] is a normalized structure, in which relationship types are not defined as fields in a record, but they are simply data values in a relationship record, thus new relationship types can be introduced with ease.																																																								
Example	The AGROVOC Thesaurus is an structured and controlled vocabulary designed to cover the terminology of all subject fields in agriculture, forestry, fisheries, food and related domains. This thesaurus is available at http://www.fao.org/agrovoc/ .																																																								
Graphical Representation																																																									
General	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="2" style="background-color: black; color: white;">(1) Term Entity</th> <th colspan="3" style="background-color: black; color: white;">(2) Term-Term Relationship Entity</th> <th colspan="3" style="background-color: black; color: white;">(3) Relationship Entity</th> </tr> <tr> <th style="background-color: black; color: white;">TermCode</th> <th style="background-color: black; color: white;">Term</th> <th style="background-color: black; color: white;">TermCode1</th> <th style="background-color: black; color: white;">TermCode2</th> <th style="background-color: black; color: white;">RelID</th> <th style="background-color: black; color: white;">RelID</th> <th style="background-color: black; color: white;">RelDesc</th> <th style="background-color: black; color: white;">RelAbr</th> </tr> </thead> <tbody> <tr> <td>1001</td> <td>Term1</td> <td>1001</td> <td>1003</td> <td>10</td> <td>10</td> <td>Broader Term</td> <td>BT</td> </tr> <tr> <td>1002</td> <td>Term2</td> <td>1003</td> <td>1004</td> <td>20</td> <td>30</td> <td>Related Term</td> <td>RT</td> </tr> <tr> <td>1003</td> <td>Term3</td> <td>1002</td> <td>1005</td> <td>10</td> <td>20</td> <td>Used For</td> <td>UF</td> </tr> <tr> <td>1004</td> <td>Term4</td> <td>1003</td> <td>1005</td> <td>30</td> <td></td> <td></td> <td></td> </tr> <tr> <td>1005</td> <td>Term5</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>	(1) Term Entity		(2) Term-Term Relationship Entity			(3) Relationship Entity			TermCode	Term	TermCode1	TermCode2	RelID	RelID	RelDesc	RelAbr	1001	Term1	1001	1003	10	10	Broader Term	BT	1002	Term2	1003	1004	20	30	Related Term	RT	1003	Term3	1002	1005	10	20	Used For	UF	1004	Term4	1003	1005	30				1005	Term5						
(1) Term Entity		(2) Term-Term Relationship Entity			(3) Relationship Entity																																																				
TermCode	Term	TermCode1	TermCode2	RelID	RelID	RelDesc	RelAbr																																																		
1001	Term1	1001	1003	10	10	Broader Term	BT																																																		
1002	Term2	1003	1004	20	30	Related Term	RT																																																		
1003	Term3	1002	1005	10	20	Used For	UF																																																		
1004	Term4	1003	1005	30																																																					
1005	Term5																																																								

Table 4.13: Pattern for re-engineering a term-based thesaurus, which follows the relation-based model, into an ontology (continued)

Slot	Value																																																								
Example	<table border="1"> <thead> <tr> <th colspan="2">(1) agrovocterm</th> <th colspan="3">(2) termlink</th> <th colspan="3">(3) linktype</th> </tr> <tr> <th>TermCode</th> <th>Term</th> <th>TermCode1</th> <th>TermCode2</th> <th>LinktypeID</th> <th>LinktypeID</th> <th>LinkDesc</th> <th>LinkAbr</th> </tr> </thead> <tbody> <tr> <td>1328</td> <td>Paddy</td> <td>5435</td> <td>6599</td> <td>90</td> <td>50</td> <td>Broader Term</td> <td>BT</td> </tr> <tr> <td>1474</td> <td>Cereals</td> <td>5435</td> <td>3354</td> <td>50</td> <td>90</td> <td>Related Term</td> <td>RT</td> </tr> <tr> <td>3354</td> <td>Poaceae</td> <td>6599</td> <td>5435</td> <td>90</td> <td>60</td> <td>Narrower Term</td> <td>NT</td> </tr> <tr> <td>5435</td> <td>Oryza</td> <td>6599</td> <td>1474</td> <td>50</td> <td>20</td> <td>Used For</td> <td>UF</td> </tr> <tr> <td>6599</td> <td>Rice</td> <td>6599</td> <td>1328</td> <td>20</td> <td></td> <td></td> <td></td> </tr> </tbody> </table>	(1) agrovocterm		(2) termlink			(3) linktype			TermCode	Term	TermCode1	TermCode2	LinktypeID	LinktypeID	LinkDesc	LinkAbr	1328	Paddy	5435	6599	90	50	Broader Term	BT	1474	Cereals	5435	3354	50	90	Related Term	RT	3354	Poaceae	6599	5435	90	60	Narrower Term	NT	5435	Oryza	6599	1474	50	20	Used For	UF	6599	Rice	6599	1328	20			
(1) agrovocterm		(2) termlink			(3) linktype																																																				
TermCode	Term	TermCode1	TermCode2	LinktypeID	LinktypeID	LinkDesc	LinkAbr																																																		
1328	Paddy	5435	6599	90	50	Broader Term	BT																																																		
1474	Cereals	5435	3354	50	90	Related Term	RT																																																		
3354	Poaceae	6599	5435	90	60	Narrower Term	NT																																																		
5435	Oryza	6599	1474	50	20	Used For	UF																																																		
6599	Rice	6599	1328	20																																																					
OUTPUT: Designed Ontology																																																									
General	<p>The ontology generated will be based on the lightweight ontology architectural pattern (AP-LW-01)[SFBG⁺07]. The thesaurus Term, schema component, will be transformed to a class, the hierarchical relationship will be transformed either to a <i>subClassOf</i> or <i>partOf</i> relation, the associative relationship will be transformed to an <i>ad-hoc</i> relation, and equivalent terms, the ones from the USE relationships, will be transformed to labels, by using the logical pattern proposed by Corcho et al. [CR09]. Finally, the content of the thesaurus will be transformed into ontology instances.</p>																																																								
Graphical Representation																																																									
(UML) General Solution Ontology	<pre> classDiagram class PreferredTerm { type PreferredTerm } class Term3 { -rdfs:label Term4 } class Term1 class Term5 class Term2 PreferredTerm < -- Term3 PreferredTerm < -- Term1 PreferredTerm < -- Term5 PreferredTerm < -- Term2 </pre>																																																								
(UML) Example Solution Ontology	<pre> classDiagram class AgrovocTerm { type AgrovocTerm } class Poaceae class Cereals class Oryza class Rice { -rdfs:label Paddy } AgrovocTerm < -- Poaceae AgrovocTerm < -- Cereals AgrovocTerm < -- Oryza AgrovocTerm < -- Rice </pre>																																																								
PROCESS: How to Re-engineer																																																									
General	<ol style="list-style-type: none"> 1. Create a class for the <i>PreferredTerm</i> schema component. 2. Assert that the <i>PreferredTerm</i> class is <i>subClassOf</i> itself, by using the proper logical pattern. 3. Assert that the <i>PreferredTerm</i> class is <i>relatedTo</i> itself. 4. Select all the preferred terms, by using the relation-based data model information. 5. For each one of the preferred terms t_i: <ol style="list-style-type: none"> 5.1. Create the corresponding ontology instance it_i. 5.2. Assert that it_i is instance of the <i>PreferredTerm</i> class. 5.3. Identify the thesaurus terms, t_q, which are equivalent terms of t_i. They are referenced in the same record which contains t_i. 5.4. For each one of the above identified thesaurus terms t_q: <ol style="list-style-type: none"> 5.4.1. Use the logical pattern proposed by Corcho et al. [CR09]. 																																																								

Table 4.13: Pattern for re-engineering a term-based thesaurus, which follows the relation-based model, into an ontology (continued)

Slot	Value
Example	<ol style="list-style-type: none"> 1. Create a <code>AgrovocTerm</code> class for the <i>PreferredTerm</i> schema component. 2. Assert that the <code>AgrovocTerm</code> class is <i>subClassOf</i> itself, by using the proper logical pattern. 3. Assert that the <code>AgrovocTerm</code> class is <i>relatedTo</i> itself. 4. Create the <code>Poaceae</code> instance and assert that it is instance of the <code>AgrovocTerm</code> class. 5. Create the <code>Oryza development</code> instance and assert that it is instance of the <code>AgrovocTerm</code> class. 6. Create the <code>Oryza</code> instance and assert that it is instance of the <code>AgrovocTerm</code> class. 7. Create the <code>Cereals</code> instance and assert that it is instance of the <code>AgrovocTerm</code> class. 8. Assert that <code>Rice</code> is label of the <code>Paddy</code> class.
Formal Transformation	
General	Thesaurus: $T = \langle TS, TD \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $TS \rightarrow OS$ $TD \rightarrow KB$
Relationships	
Relations to other modelling components	Use the Architectural Pattern: AP-LW-01 [SFBG ⁺ 07]

Chapter 5

Evaluation of the Method and Patterns for Re-engineering Non-Ontological Resources

In this chapter we provide qualitative and (where possible) quantitative evidence that using the method, the set of patterns, and the PR-NOR software library, leads to users being able to design ontologies faster and/or better quality standards, in other words, to assess how effective the method, the set of patterns, and the PR-NOR software library are for the target user.

The next set of experimentation and findings concerns those user studies with the technologies deployed as the PR-NOR software library, that implements the patterns guidelines for transforming non-ontological resources into ontologies. These studies focus on testing the overall performance and design suitability of the implemented techniques and the effectiveness of the methodological components.

5.1 Supporting Non-ontological Resource Re-engineering

One of the process identified in the ontology network development is the Non-Ontological Resource Re-engineering process. The Neon Glossary of processes and activities [SFdCB⁺07] defines Non-Ontological Resource Re-engineering as the process of taking an existing non-ontological resource and transforming it into an ontology. Chapter 4 describes in detail the methodological guidelines for supporting this process. Our concern here is to describe a set of experiments that can show the effects and benefits of using Non-Ontological Resource Re-engineering guidelines. There are several different aspects of guidelines that need to be studied and several types of effects of guidelines usage that need to be defined and measured.

5.1.1 Overview and Objectives

Methodology experts participating in the NeOn project have identified Non-Ontological Resource Re-engineering as one of the crucial processes in the ontology network development. In this deliverable, we propose an experiment to learn about the understandability and usability of the methodological guidelines for carrying out the Non-Ontological Resource Re-engineering Process. The main goal of the experiment is to test the benefits of using the proposed methodological guidelines and additional material included in D5.4.2 [SFdCB⁺09], D2.2.2 [VTAGS⁺08], and chapter 4 of this deliverable, for speeding up the development of ontologies.

5.1.2 Assumptions and user study setup

In this experiment we worked with a questionnaire about the methodological guidelines for the non-ontological resource re-engineering process, to be answered by people carrying out the experiment. People carrying out the experiment have different experience levels and background in databases, software engineering, etc, and some experience in ontology engineering.

5.1.3 Finding and observations

The experiment was carried out in

- The "Ontologies and Semantic Web" course from the "Athens Programme" at the Facultad de Informática (Universidad Politécnica de Madrid) with master students, having background in databases, software engineering, artificial intelligence, and some experience in ontology engineering.
- The "Ontologies and Semantic Web" course from the "Information Technology" master at the Facultad de Informática (Universidad Politécnica de Madrid) with master students, having background in databases, software engineering, artificial intelligence, and some experience in ontology engineering.

We proposed a questionnaire about the use of methodological guidelines for non-ontological resource re-engineering process. Section 4.2 shows the workflow corresponding to such guidelines. For interpreting the results, we analyzed the answered questionnaires and extracted some statistics.

The questionnaire includes the following questions:

1. Are the proposed guidelines well explained?
2. Is more detail needed in the guidelines? If so, please explain in detail in which sense and in which tasks.
3. Do you think more techniques and tools should be provided?
4. How can we improve the proposed guidelines? In which tasks?
5. Did you find these guidelines useful?

The experiment is divided in the following phases:

1. Lecture providing to students the proposed guidelines included in deliverable D5.4.2, D2.2.2. and section 4.2 of this deliverable.
2. Student groups analyze the methodological guidelines proposed to carry out the non-ontological resource re-engineering process.
3. Students fill in a questionnaire about the proposed guidelines.

5.1.4 Further analysis and discussion

The experiment included 5 questions about the methodological guidelines solved by 26 students, and as a general conclusion we can say that students did not have problems with the use and understanding of each one of the activities and tasks identified in the methodological guidelines. In the following, we provide some observations extracted from the analysis of the experiment results:

- 97% of the comments provided by the students to question 1 indicated that guidelines were well explained.
- For the comments obtained for question 2: "Is more detail needed in the guidelines?", we can say that 88% of the students consider that more detail is not necessary in the guidelines, however 12% think there is an opportunity to improve the explanations of: i) how to search for a suitable pattern (task 4 in the guidelines), and ii) how to perform the ontology formalization (task 7 in the guidelines).
- For the comments obtained to question 3: "Do you think more techniques and tools should be provided?", we can say that all evaluators believe that the techniques and tools to execute each activity of the guidelines are sufficient.

- The generalized comment to question 4 "How can we improve the proposed guidelines?" is to include more examples of how to use the proposed guidelines for the non-ontological resources re-engineering process and what results are expected.
- Finally, with respect to question 5 "Did you find these guidelines useful?", all students believed that the guidelines were useful, but also necessary.

One of the objectives of the research on non-ontological resource re-engineering process should be to improve the detail and completeness of the guidelines, as suggested by the responses to the questions two and three and for this reason some actions are proposed in this deliverable (see Section 4.2).

Identified strengths and weaknesses

Based on the comments obtained in this experiment we can say that the majority of students found that the methodological guidelines were useful and understandable. The main weaknesses included a more complete description of some tasks of the methodology. Some examples are:

- A more detailed description of the criteria to search for a suitable pattern.
- More examples of how to use the guidelines. Moreover, we are studying the possibility of including more examples of test cases solved by the guidelines.

Prospective further work

Based on the analysis carried out with the data extracted from the questionnaires, we are currently working on:

- Including more detail in the third activity of the methodological guidelines.
- Adding a new task to check if some inconsistency is present after the transformation.

5.2 Testing the PR-NOR Software Library

This reported study refers to testing the adequacy and usability of the PR-NOR software library in the context of the development of ontologies. The main motivation of this experiment was to evaluate the PR-NOR software library. The PR-NOR software library is described in deliverable *D2.5.2 Pattern based ontology design: methodology and software support*.

5.2.1 Overview and Objectives

The main purpose and aim of the study is to determine if the set of patterns really make ontology development easier and faster. Additionally, we study the implementation support, the PR-NOR software library.

The goal of the first experiment is to evaluate the time and effort spent, and the quality of the ontologies developed manually against the ontologies constructed using the PR-NOR software library. The non-ontological resource utilized in this experiment is a classification scheme.

The goal of the second experiment is also to evaluate the time and effort spent, and the quality of the ontologies developed manually against the ontologies constructed using the PR-NOR software library. In this experiment the non-ontological resource utilized is a thesaurus.

The goal of the third experiment is to compare the quality of the ontologies constructed from a classification scheme and a thesaurus. To this end, the experiment utilizes a gold standard ontology. The participants

compute the similarity between the ontology constructed from a classification scheme and the gold standard ontology, and the similarity between the ontology constructed from a thesaurus and the gold standard ontology.

The goal of the fourth experiment is to assess the user satisfaction of the PR-NOR software library for carrying out the non-ontological resource re-engineering process.

5.2.2 Assumptions and user study setup

The first, second and fourth experiment were carried out in

- The "Ontologies and Semantic Web" course from the "Athens Programme" at the Facultad de Informática (Universidad Politécnica de Madrid) with master students, having background in databases, software engineering, artificial intelligence, and some experience in ontology engineering.
- The "Ontologies and Semantic Web" course from the "Information Technology" master at the Facultad de Informática (Universidad Politécnica de Madrid) with master students, having background in databases, software engineering, artificial intelligence, and some experience in ontology engineering.

The third and fourth experiment were carried out in the "Ontologies and Semantic Web" course from the "Artificial Intelligence (AI)" master at the Facultad de Informática (Universidad Politécnica de Madrid) with master students, having background in databases, software engineering, and artificial intelligence, with some practical experience in ontology engineering.

User study 1: Time/effort spent, and quality of the resultant ontology, by using a classification scheme

For this experiment we selected a classification scheme of a given domain implemented in a database. The experiment was divided in the following phases:

- Student groups build a conceptual model manually, of the given domain, from scratch.
- Student groups build other ontology using the PR-NOR software library.
- Student groups compare the time, effort and quality of the resultant ontologies.
- Student groups fill in a questionnaire about the time, effort and quality of the resultant ontologies.

The questionnaire, that allows us to collect the assessments of the students about the time, effort and the quality of the ontology, consists of the following questions:

1. How much time did you spend for building the conceptual model?
2. How much time did you spend for building the ontology using the PR-NOR software library?
3. In your opinion which of the two resultant ontologies has best degree of quality?

User study 2: Time/effort spent, and quality of the resultant ontology, by using a thesaurus

For this experiment we selected a thesaurus of a given domain implemented in an XML file. The experiment was divided in the following phases:

- Student groups build a conceptual model manually, of the given domain, from scratch.
- Student groups build other ontology using the PR-NOR software library.

- Student groups compare the time, effort and quality of the resultant ontologies.
- Student groups fill in a questionnaire about the time, effort and quality of the resultant ontologies.

The questionnaire, that allows us collecting the assessments of the students about the time, effort and the quality of the ontology, is the same of the user study 1.

User study 3: Quality of the resultant ontology

For this experiment we selected a domain. The experiment was divided in the following phases:

- We provide to the student groups an ontology, a classification scheme and a thesaurus.
- Student groups refine the given ontology.
- Student groups build the ontology using the PR-NOR software library from the selected thesaurus.
- Student groups build the ontology using the PR-NOR software library from the selected classification scheme.
- Student groups compare the resultant ontologies using a similarity measure based on Cider System¹.
- Student groups fill in a questionnaire about the similarity among the ontologies.

The questionnaire, that allows us collecting the assessments of the students about the similarity of the ontology, consists of the following questions:

1. In your opinion which of the three ontologies has best degree of quality? Why?
2. Please provide the similarity degree between the given ontology and the one constructed from the classification scheme.
3. Please provide the similarity degree between the given ontology and the one constructed from the thesaurus.
4. Please comment how the non-ontological resource type (i.e. classification scheme and thesaurus) influences the resultant ontology.

User study 4: User Satisfaction

For this purpose we conducted an experiment following the Software Usability Measurement Inventory (SUMI) method [KC93]. The SUMI questionnaire includes 50 items for which the user selects one of three responses ("agree", "undecided", "disagree"). The following sample shows the kind of questions that were asked:

- This software responds too slowly to inputs.
- I would recommend this software to my colleagues.
- The instructions and prompts are helpful.
- I sometimes wonder if I am using the right command.
- Working with this software is satisfactory.

¹<http://sid.cps.unizar.es/SEMANTICWEB/ALIGNMENT/>

- I think that this software is consistent.

The experimenters met with all participants for 10 minutes to explain the purpose of the evaluation session and present the methodology of SUMI evaluation. Then, participants had 20 minutes to test the PR-NOR software library, and 10 minutes to fill the SUMI questionnaire for user-interaction satisfaction. During these two phases of the experiment users were not allowed to ask questions to the experimenters. The questionnaire was designed to measure the affect, efficiency, learnability, helpfulness and control [DR93]. SUMI is also mentioned in the ISO 9241 standard as a recognized method of testing user satisfaction [ISO98].

5.3 Findings and observations

In this section we provide some findings extracted from the analysis of the experiment results.

5.3.1 User study 1: Time/effort spent, and quality of the resultant ontology, by using a classification scheme

From this experiment, in which students compare the time, effort and quality of ontologies built manually against ontologies built using the PR-NOR software library for transforming a classification scheme, we can mention the following observations:

- As Figure 5.1 shows, 98% percent of the students identified more effort and time invested for the ontologies developed manually than the ontologies developed using the PR-NOR software library.

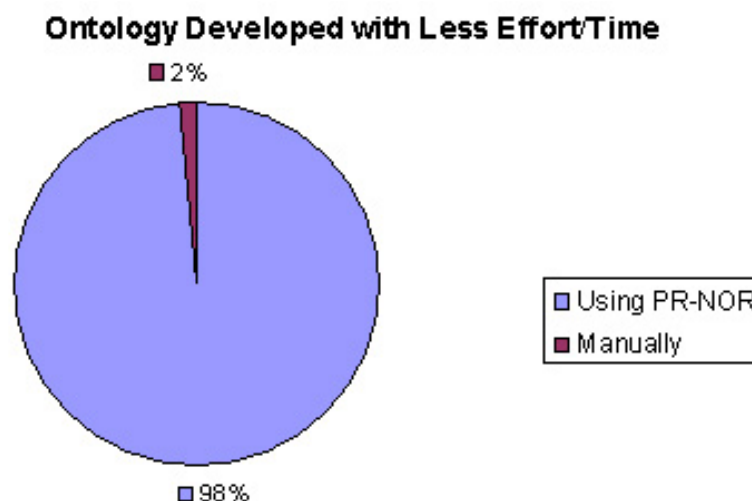


Figure 5.1: Personal assessment about the ontology developed with less effort/time

- As Figure 5.2 shows, 92% percent of the students have identified the ontology constructed using the PR-NOR software library 'better' in some modelling quality sense, than the ontology constructed manually. 4% of the students were not sure about which one had the better quality, and the rest of the students believed that the ontology constructed manually has better quality than the ontology constructed using the PR-NOR software library.

5.3.2 User study 2: Time/effort spent, and quality of the resultant ontology, by using a thesaurus

From this experiment, in which students compare the time, effort and quality of ontologies built manually against ontologies built using the PR-NOR software library for transforming a thesaurus, we can mention the

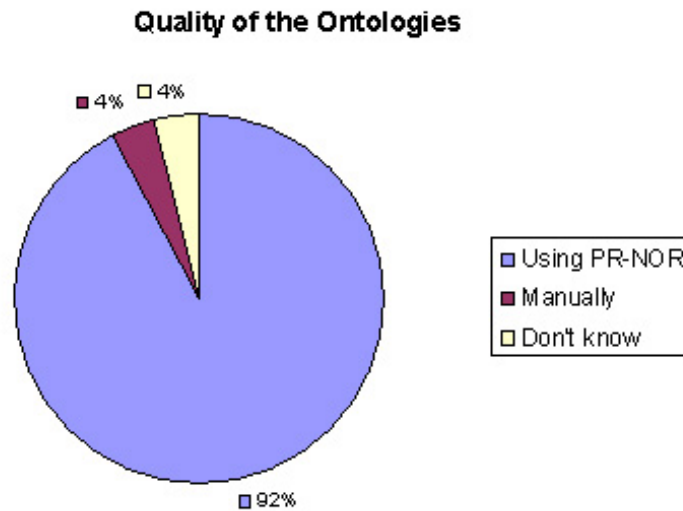


Figure 5.2: Personal assessment about the quality of the ontologies

following observations:

- As Figure 5.3 shows, 98% percent of the students identified more effort and time invested for the ontologies developed manually than the ontologies developed using the PR-NOR software library.

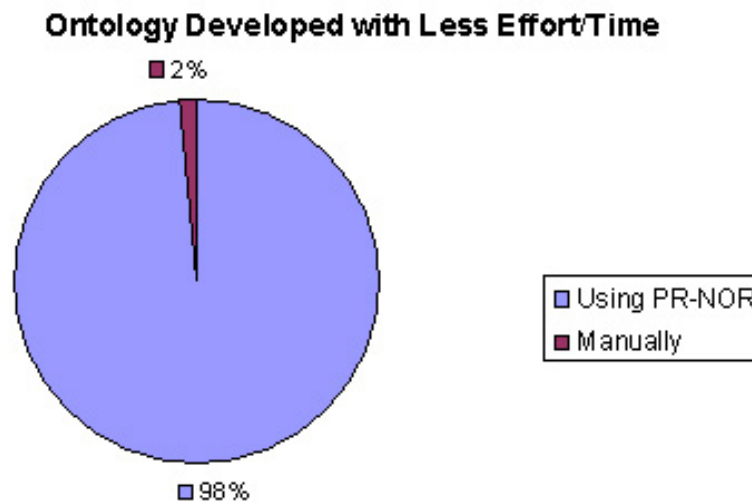


Figure 5.3: Personal assessment about the ontology developed with less effort/time

- As Figure 5.4 shows, 84% percent of the students have identified the ontology constructed using the PR-NOR software library 'better' in some modelling quality sense, than the ontology constructed manually. 4% of the students were not sure about which one had the better quality, and the rest of the students believed that the ontology constructed manually has better quality than the ontology constructed using the PR-NOR software libraries.

5.3.3 User study 3: Quality of the resultant ontology

From this experiment, in which students compare the quality of the ontology constructed from the classification scheme against the ontology constructed from the thesaurus, we can mention the following observations:

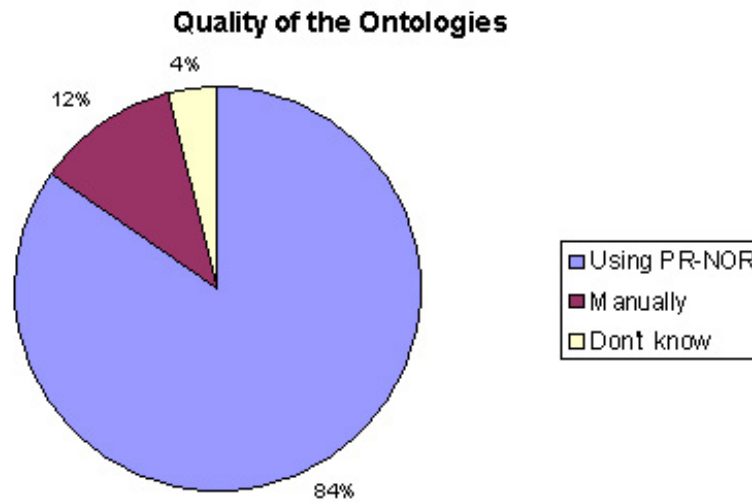


Figure 5.4: Personal assessment about the quality of the ontologies

- As Figure 5.5 shows, 74% percent of the students have identified the ontology constructed from the classification scheme 'better' in some modelling quality sense, than the ontology constructed from the thesaurus.

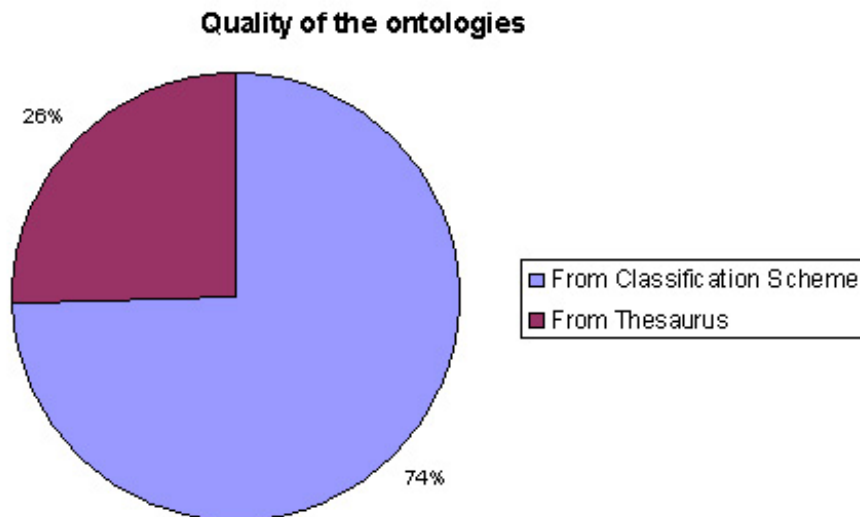


Figure 5.5: Personal assessment about the quality of the ontologies

5.3.4 User study 4: User Satisfaction

As a general conclusion we can say that the results of the evaluation are very positive. The analysis of the results of the experiment that we conducted shows several good points of the PR-NOR software library as well as some issues that could be improved as part of our future work. Figure 5.6

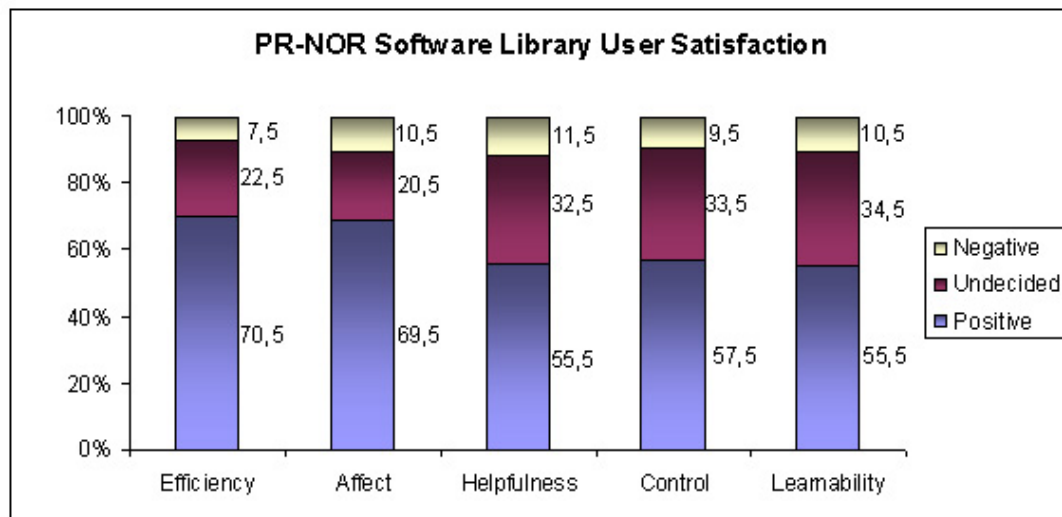


Figure 5.6: The results of SUMI questionnaires for the PR-NOR Software Library

5.4 Further analysis and discussion

In this section we include the analysis of the four user studies carried out with the PR-NOR software library.

5.4.1 Analysis of the user study 1

In this section we analyze the questionnaire filled in by the group of students that carried out the experiment. Regarding the time/effort spent, and quality of the resultant ontology by using a classification scheme, these are the main conclusions:

- PR-NOR software library really makes ontology development easier and faster. So, the user saves time and effort in the development of ontologies.
- PR-NOR software library generates ontologies with an acceptable level of quality. This is because we are relying on external resources to disambiguate the relations among the non-ontological resource entities (see section 4.3.3).

5.4.2 Analysis of the user study 2

In this section we analyze the questionnaire filled in by the group of students that carried out the experiment. Regarding the time/effort spent, and quality of the resultant ontology by using a thesaurus, these are the main conclusions:

- PR-NOR software library really makes ontology development easier and faster. So, the user saves time and effort in the development of ontologies.
- PR-NOR software library generates ontologies with an acceptable level of quality. This is because we are relying on external resources for disambiguate the relations among the non-ontological resource entities (see section 4.3.3).

5.4.3 Analysis of the user study 3

In this section we analyze the questionnaire filled in by the group of students that carried out the experiment. Regarding the quality of the resultant ontology, these are the main conclusions:

- PR-NOR software library generates ontologies with an acceptable level of quality.
- The quality of the ontology constructed is influenced by the non-ontological resource type. Apparently the ontologies constructed from a classification scheme are 'better' in some modelling quality sense, than the ontologies constructed from a thesaurus.

5.4.4 Analysis of the user study 4

In this section we describe the results obtained for each dimension of SUMI questionnaire:

Efficiency

The higher value was obtained for this dimension, therefore we believe that the evaluation of the efficiency of the PR-NOR software library is satisfactory.

Affect

The affect dimension measures the user's general emotional reaction to the software, it may be called "Likeability". For this dimension we found that the question that most contributed to 10.5% of disagreement in the user's general reaction to the software was: "I feel safer if I use only a few familiar commands or operations". We believe that we must improve this aspect of the PR-NOR software library, so that all functionalities can be perceived with same degree of positivity by users.

Helpfulness

55.5% of the users believe that the software is self-explanatory (helpful). Moreover, we found that the question that more contributed to 32.5% of indecision was: "This software is awkward when I want to do something which is not standard". This means that the majority of the users did not have the need to find alternative options to perform the available actions in the software library.

Control

We consider that the evaluation of the degree to which the user feels that (s)he, and not the product, is setting the pace, is satisfactory, because we only obtained 9.5% disagreement. In the same sense, 33.5% of indecision, correspond to aspects that did not appear in the software such as "The software allows the user to be economic of keystrokes", which is positive.

Learnability

The lowest value was obtained for this dimension, therefore we believe that the evaluation of the learnability of the software library is not good. We believe that we must improve this aspect of the PR-NOR software library, so that the speed and facility with which the users feel that they learn how to use new features when necessary, will increase.

5.4.5 Identified strengths and weaknesses

According to the results from the first and second experiment, the main strength of the PR-NOR software library is the effort and time saved for building ontologies.

According to the results from the third experiment, the main strength of the PR-NOR software library is the quality of the ontologies constructed. However, this quality is influenced by the non-ontological resource type.

The results show that the ontologies constructed from a classification scheme are 'better' in some modelling quality sense, than the ontologies constructed from a thesaurus.

We have to improve the quality of the ontologies constructed from a thesaurus, by relying on more trusted external resources to discover/disambiguate the relations among the non-ontological resource entities.

Based on the comments obtained in experiment 4 we can say that the majority of students found the experience with PR-NOR software library satisfactory. However, it is necessary to improve the helpfulness and learnability aspects, as suggested by the responses to the questions on these dimensions.

5.4.6 Prospective further work

As further work we are applying a follow-up study to investigate precisely both quality and impact of the external resources used for discover/disambiguate the relations among the non-ontological resource entities. We are planning to include DBpedia² as an external resource.

Regarding the user satisfaction, we are planning to develop a GUI for the PR-NOR software library. In this way the helpfulness and learnability aspects will be improved for the resultant system.

²DBpedia is a community effort to extract structured information from Wikipedia and to make this information available on the Web. <http://dbpedia.org/About>

Chapter 6

Improvements of the Folksonomy Enrichment Algorithm

In this chapter we present the finalisation of the method described in [SAd⁺07b] and [VTAGS⁺08] on Folksonomy Enrichment. In the above deliverables we demonstrated that the **automatic enrichment of folksonomy tagsets using a combination of Knowledge Sources (KS) such as WordNet and online ontologies is possible** without user intervention in any step of the process and by using straightforward methods for lexical isolation, disambiguation, semantic expansion and semantic enrichment. In this chapter we present an improved version of this work. More specifically we describe:

1. A comparative study on the Knowledge Sources (KS) used for the enrichment.
2. The improved version of the FLOR enrichment algorithm

6.1 Introduction

In this section we present some basic concepts and the overview of this chapter.

6.1.1 Background and Motivation

Folksonomies are a convenient medium to publish, annotate and share content on the web. Their basic entities are the **users**, who annotate (i.e., tag) **resources** with **tags** (text labels). Figure 6.1a shows a snippet of an example folksonomy where resources (R_1 , R_2 and R_3) are tagged with a number of tags.

Definition 1. A folksonomy consists of a set of resources $R = \{r_1, \dots, r_{|R|}\}$ and a set of tags $T = \{t_1, \dots, t_{|T|}\}$. For a resource $r \in R$, $tags(r)$ represents its set of tags, its **TagSpace**. For a tag $t \in T$, $res(t) = \bigcup_{i=1}^{|R|} r_i \forall r_i : t \in tags(r_i)$ is the set of resources tagged with tag t . For example, in Figure 6.1a, $tags(R_1) = \{fruit, dessert\}$ and $res(apple) = \{R_2, R_3\}$.

Due to the lack of tagging restrictions the following phenomena have been observed [GH06], which hamper the process of search in folksonomies:

- **Tag synonymy** arises when lexically different tags express the same concept, e.g., *cake* and *dessert*. Synonymy may cause exclusion of results if these are tagged with synonym(s) of the search keyword, e.g., in Figure 6.1a, searching for *cake* will return only R_2 and not R_1 .
- **Basic level variation.** Tags with different levels of specificity are used to describe resources that relate to the same concept. For example, *apple* and *fruit* can both describe resources about apples. The **lack of structure** in tagspaces, does not allow for explicit declaration of the fact "*apple is a fruit*". This

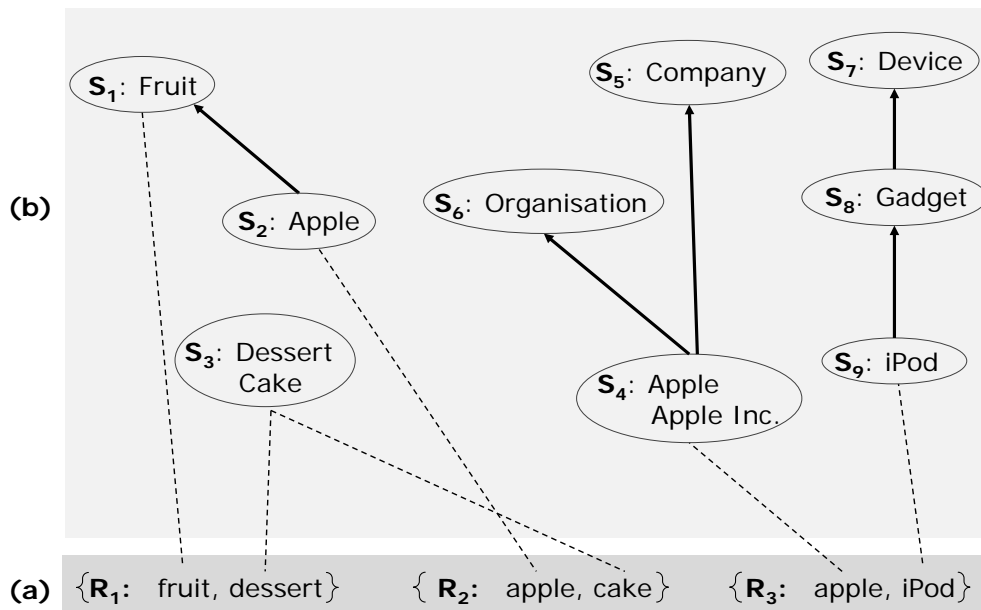


Figure 6.1: Part (a) represents a folksonomy tag space (tags are textual labels). Part (b) represents the semantic layer over this tag space as a result of the folksonomy enrichment

limits the potential of querying for resources tagged with related tags. For example, for the folksonomy of Figure 6.1a, querying for *fruit* only returns R_1 , although R_2 is also tagged with a fruit. Moreover, the lack of structure does not allow for result diversity. Result diversity [vZMPR08] is the grouping of similar results in distinctive sets (e.g., querying for fruit returns one set with apples, one for oranges e.t.c.).

- **Tag polysemy** occurs among lexically identical tags that denote different meanings. For example, *apple* may refer to *fruit* or a *company*. Tag polysemy causes the retrieval of unwanted results when the tag is used with a different meaning than the search keyword, e.g., searching for *apple* in the folksonomy of Figure 6.1a will return both R_2 and R_3 , although, depending on the meaning of *apple* in the context of the query, only one of these resources is relevant.

The objective of folksonomy re-engineering, folksonomy enrichment, is the transformation of flat folksonomies to rich semantic structures by re-using publicly available **Knowledge Sources (KS)**. These semantic structures assign meaning to the folksonomy tags by re-using **Semantic Entities (SE)** located either in ontologies or in WordNet. In addition, the relations among the SEs are known and as a result, the relations among the tags are made explicit. Furthermore, SEs on the Semantic Web or in WordNet are rich in semantic and lexical information (e.g., related SEs and further textual descriptions such as labels or comments). This information is integrated into the **Semantic Layer (SL)** produced from the folksonomy enrichment method (exemplified in Figure 6.1b) resulting to a network of ontologies and folksonomy resources.

Such a SL representing a folksonomy tag space contains the required information to solve polysemy (e.g., *fruit* \neq *company*) and synonymy (e.g., *cake* = *dessert*) issues, and to allow for query expansion based on the concept relations. It also offers the possibility to present the results in an intuitive way allowing for result diversity as well as facilitate intelligent search, navigation, annotation as well as the integration of resources from distinct, heterogeneous Web2.0 systems.

6.1.2 Overview

In [VTAGS⁺08] we presented a version of the FLOR enrichment algorithm and the experiments we performed to test the hypothesis that **enrichment of folksonomy tagsets with ontological entities can be performed automatically**. As indicated by the results, the quality and performance of the enrichment output depended on the disambiguation algorithms of FLOR and the (KS) used for the enrichment purposes, namely WordNet and ontologies. In addition, most of the efforts towards combining folksonomies with semantics have employed either ontologies or WordNet. These two facts motivated a comparative study of WordNet and ontology usage, presented in Section 6.2.

Section 6.3 presents the enhancements in the FLOR algorithm compared the version reported in [VTAGS⁺08] as well an additional FLOR sub-procedure that produces an integrated SL (in the last version of the algorithm, the integration among the sense of the tags was not implemented).

6.2 Evaluation of Knowledge Sources

WordNet is a long term, continuously maturing project used for information retrieval, text classification and sense disambiguation and spans over several knowledge domains. As an initiative of a closed research team, WordNet's evolution depends on this team and may be slow. However, due to the same reason it is of high quality and contains limited errors. Some of the works exploiting WordNet to resolve folksonomies problems are [LY07, LEC07].

An alternative paradigm utilises **(online) ontologies** to improve folksonomies. Different types of ontologies exist, varying in scope and context as they are built to serve the purposes of specific tasks, applications or describe certain domains. Because of that, they cover different knowledge domains in variable levels of detail. Ontologies are more recent than WordNet and contain new terms due to the fact that they are created and updated by many knowledge experts. As a result of this, the knowledge defined in ontologies is diverse and at times redundant. Furthermore, due to the lack of quality control, ontologies suffer from modelling errors (e.g. *China subclassOf Asia*) as described in previous work [SGA⁺07].

The lack of comparative studies on WordNet and ontologies motivated our interest to explore these two KS in terms of folksonomy search. Our goal in this evaluation is twofold. First, we investigate **how the usage of KS can address polysemy, synonymy and basic level variation and minimise their effects on folksonomy search**. Second we **compare the KS-based search with the cluster-based search in folksonomies**. We present a method that creates a naive structure of senses using one KS at a time (Sec.6.2.1). We implement a query expansion mechanism (Sec.6.2.1) for these structures and using a web interface we compare the KS and the cluster-based search (Sec.6.2.2) within a user study.

6.2.1 Building and Querying Knowledge Structures

The goal of this work is to compare the characteristics of WordNet and ontologies that contribute to the creation of semantic structures (e.g., Figure 6.1b) and to the resolution of the search impediments caused by polysemy, synonymy and basic level variation. Although context investigation and disambiguation mechanisms are required in order to define the precise meaning and relations of tags, the analysis of disambiguation techniques is bound to be influenced from the selection of the KS and as a result is eliminated from this study. In this work we focus on studying which characteristics of the KS can deal with the above phenomena. We start by defining the basic elements of our approach.

Definition 2. A Knowledge Source *KS* consists of a set of senses $S = \{s_1, \dots, s_{|S|}\}$. Each sense *s* has a number of more specific, *subSenses* $sub(s)$, and more generic, *superSenses* $sup(s)$, senses. For example, in Figure 6.1b $sub(S_1) = \{S_2\}$. In addition, a sense *s* has a set of words that define its meaning, its set of synonyms, and it is denoted by *syn*(*s*). For example, in Figure 6.1b $syn(S_3) = \{cake, dessert\}$.

Definition 3. We define the relation between a tag $t \in T$ and a sense $s \in S$ as $Dfn(t, s)$, $\forall t, \forall s : t \in syn(s)$. This means that t is potentially defined by s if it belongs to its set of synonyms. In addition, $senses(t) = \bigcup_{i=0}^{|S|} s_i : Dfn(t, s_i)$ is the set of senses assigned by the KS to the tags representing its possible meanings. For example, in Figure 6.1b $senses(apple) = \{S_2, S_4\}$.

As previously mentioned, in Figure 6.1a, searching for *cake* excludes R_1 due to **tag synonymy**. $Query Results = res(cake) = R_2$. The existence of a semantic structure allows for the inclusion of results tagged with synonym tags to *cake*, i.e., $Query Results = \bigcup_{i=0}^T res(t_i) : t_i \in syn(s_j) \forall s_j : Dfn(cake, s_j) = \{R_1, R_2\}$. Intuitively, the richer a sense s in terms of synonyms, the higher the probability of retrieving resources tagged with a tag that is equivalent to a synonym. Therefore, to compare the two KS in terms of dealing with tag synonymy we measure the average number of synonyms per sense assigned to tags: (I) $\overline{|syn(S)|} = \frac{\sum_{i=0}^{|S|} |syn(s_i)|}{|S|}$. For example, in Figure 6.1b $\overline{|syn(S)|} = \frac{1+1+2+2+1}{5} = 1.4$

In the same scenario, searching for *apple* retrieves both R_2 and R_3 due to **tag polysemy**. To allow for querying mechanisms to deal with polysemy the structure should have enough senses representing the meanings of polysemous tags. The more senses a KS provides for a tag the more likely to match all the possible senses of the tag. Therefore, we compare the two KS in terms of dealing with tag polysemy by measuring the average number of senses per tag: (II) $\overline{|senses(T)|} = \frac{\sum_{i=0}^{|T|} |senses(t_i)|}{|T|}$. E.g., in Figure 6.1 $\overline{|senses(T)|} = \frac{1+1+2+1+1}{5} = 1.2$

To deal with problems caused by lack of structure and **basic level variation** we investigate how tags can be mapped onto the hierarchical structure of senses in each KS. First, we measure the mean number of direct sub/super-senses. The higher the number of sub/super-senses the higher the probability to map more specific and more generic tags of a tag into the structure. The mean numbers of subsenses and supersenses are: (III) $\overline{|sub(s)|} = \frac{\sum_{i=0}^{|S|} |sub(s_i)|}{|S|}$ and (IV) $\overline{|sup(s)|} = \frac{\sum_{i=0}^{|S|} |sup(s_i)|}{|S|}$. In Figure 6.1b $\overline{|sup(S)|} = \frac{0+1+0+2+1}{5} = 0.8$.

Formulas (I to IV) calculate the richness of the KS, however, to decide which KS performs better during search, one more measure needs to be defined. This represents how well the expansion of t using each KS is mapped to the tagspace and is reflected by the ratio of the resources retrieved using only t to the resources retrieved using the expansion of t . The expansion of t is defined as $exp(t) = \{syn(s) + syn(sub(s)) + syn(sup(s))\}, \forall s : Dfn(t, s)$. The increase ratio for tag t is defined as: $inc(t) = \frac{|res(exp(t)) - res(t)|}{|res(exp(t)) + res(t)|}$. E.g., $inc(cake) = \frac{|\{R1\} - \{R2\}|}{|\{R1\} + \{R2\}|} = \frac{1}{2} = 0.5$. We measure the mean increase for T which is: (V) $\overline{|inc(T)|} = \frac{\sum_{i=0}^{|T|} |inc(t_i)|}{|T|}$.

Finally, to obtain an estimation of how well a KS can solve the above phenomena, the set of tags covered by this KS, $T_{KS} \subseteq T$ should be measured. Intuitively, the higher the number of tags defined by the KS the better the KS performs in this task. (VI) $|T_{KS}| = |\bigcup_{i=0}^{|T|} t_i : \exists s \in S : Dfn(t_i, s)|$.

To evaluate measures (I to VI), first, we use each KS to assign senses to the tags and to link the senses with their hierarchical information extracted from each KS. This **semantic enrichment** process yields one semantic structure per KS as described in Sec.6.2.1. Second, we implement a **query mechanism** for these semantic structures as described in Sec. 6.2.1.

Step 1: Semantic Enrichment

We use two different strategies to enrich tagspaces with semantic structure. Strategy A uses WordNet and Strategy B uses online ontologies. The intended output for both strategies is a structure similar to the one depicted in Figure 6.1b. This structure is built in two stages, common to both strategies.

First, the **potential meanings of a tag** are made explicit by aligning it to appropriate senses. While in previous work [ASM08, VTAGS⁺08] we use disambiguation algorithms to precisely identify the meaning of a tag in a certain context, for the purposes of this comparative study we assign all possible senses to a tag.

As mentioned above, we are interested in the richness and coverage of the KS over a tagspace and want to rule out any bias introduced by disambiguation methods. As a result, we assign $senses(apple) = \{S_2, S_5\}$ (Figure 6.1). Strategy A relies on WordNet's synsets to find such senses, while in the case of Strategy B we developed a clustering mechanism which identifies a possible set of senses for a tag by combining information from multiple online ontologies. To estimate $|\overline{syn(S)}| (l)$ for each KS, we assign as synonyms ($syn(s)$) to each sense s all the available lexical information from the respective KS that can possibly describe the sense (e.g., synonyms from WordNet, ids and labels from ontologies).

Second, we include **structural information among the senses** by reusing knowledge from the KS. First, we select all possible ancestors for each sense, i.e., $S_4:Apple$ is defined as a $S_5:Company$ and an $S_6:Organisation$. The reason for selecting all possible ancestors is the heterogeneity of the structures of the knowledge sources. In order to achieve high connectivity among the senses we import the subsumption path to the highest possible ancestor e.g., $S_9:iPod$ is a subsense of $S_8:Gadget$ which is a subsense of $S_7:Device$. We currently restrict the method to subsumption relations, as these are present in both KS.

Strategy A: WordNet-Based Enrichment. WordNet is a hierarchy of synsets each describing a sense. Most synsets are subsumed by at least one hypernym synset; they subsume a set of hyponym synsets and contain a set of words describing the same sense (synonyms).

For **sense selection**, we consider all the synsets that contain a given tag in their list of synonyms. Note that we consider only noun synsets as these have richer hierarchical information than other parts of speech. For each sense, we import in the structure its corresponding synonyms. WordNet's matching mechanism automatically caters for lexical variations and plurals. To create a **structure of senses**, we import each sense's ancestor path till the root of the WordNet hierarchy and their first level of hyponyms.

Strategy B: Online-Ontology Based Enrichment. In order to enrich the tagspace, we explore online ontologies through the Watson¹ Semantic Web gateway. The sense selection is more difficult in this case, because, unlike WordNet, the Semantic Web does not contain an established set of senses. To overcome this limitation, we build a clustering algorithm, which groups together entities that are sufficiently similar and therefore might denote the same sense.

First, all ontological concepts containing the tag in their localname (id) or label(s) are selected. We focus on concepts as they have a richer hierarchy than properties and individuals and also to maintain comparability with the structure generated from WordNet. We use Watson's API and we strictly match the tags against the id or label(s) of ontological concepts, e.g., *berry* is not matched against *berry_fruit* neither is *water* against *water_container*. This is done to restrict additional noise.

We use the entity (sense) clustering algorithm described in Section 6.3.4 to obtain clusters of entities each of which describes a sense. Once the entity clustering is complete, for all the direct superclasses of the cluster's entities we iteratively get their superclasses until we get to the root of an ontology. For example, we obtain *Tropical Fruit* $\xrightarrow{subClassOf}$ *Fruit*. We notice that by adding this knowledge there is then one direct and one indirect relation between *Fruit* and *Banana*. We maintain as many subsumption relations as possible regardless of whether or not they are implicitly redundant in order to support query expansion.

Step 2: Query Mechanism

The query mechanism allows the exploration of the structures created by using each KS. Algorithm 1 (Alg.1) describes a querying mechanism which a) maps query keywords to appropriate senses; b) retrieves the resources tagged with tags associated to these senses, and c) groups the returned resources into meaningful groups, which are used as a basis for the presentation of the results.

Our algorithm is based on the following intuition: users are primarily interested in resources tagged with the exact keyword, as well as with tags denoting more specific concepts. However, if only a few resources are

¹<http://watson.kmi.open.ac.uk>. The ontologies indexed in Watson during the experiment (May-June 2009) were approximately 9.000 and contained a total number of 460.000 classes (including redundancies).

returned for these cases, the user might also be interested in exploring resources tagged with more generic tags. For example, when searching for *fruit*, a user is likely to also be looking for resources annotated with the various types of *fruit*, such as *apple* or *tropical fruit*. Alternatively, if few results are returned, he may be interested in broader notions e.g., *plant*. Accordingly, for a query keyword k , Alg.1 retrieves all relevant senses and all their subsenses (Alg.1, 3-5). For all these senses, it retrieves the resources that are tagged with tags mapped to these senses (Alg.1, 6-11). The algorithm also categorises all the retrieved resources into groups according to the overlap of the resources' tags with the synonyms of the subsenses. For example, in a query for *animal*, items tagged with *animal* and *zebra* and items tagged with *zebra* are grouped together into a group described with *zebra* (Alg.1, 12-18). If the number of subsenses is less than four, then the same process is repeated with the supersenses (Alg.1, 20 -27). The threshold of four is selected because we further compare the KS-based querying with the cluster-based querying (Sec.6.2.2) and the mean number of clusters per tag is 3.4.

Algorithm 1 KS-based Querying

```

1: for all  $k \in QueryKeywords$  do
2:   create group  $g_k$  ▷ This is to place all the uncategorised results
3:   retrieve  $S : \forall s \in S, Dfn(k, s)$  ▷ S=senses(k)
4:   for all  $s \in S$  do
5:     retrieve  $sub(s)$ 
6:     for all  $\acute{s} \in sub(s)$  do
7:       retrieve  $\acute{R} = \bigcup_{t \in syn(\acute{s})} res(t)$ ,
8:       create group  $\acute{g}$ 
9:        $\forall \acute{r} \in \acute{R}$ , put  $\acute{r}$  in  $\acute{g}$ 
10:    end for
11:    retrieve  $R = \bigcup_{t \in syn(s)} res(t)$ ,
12:    for all  $r \in R$  do
13:       $Overlap(r, \acute{s}) := |tags(r) \cap syn(\acute{s})| / |tags(r) \cup syn(\acute{s})|$ ,  $\acute{s} \in sub(s)$ 
14:      put  $r$  in  $\acute{g} : Overlap(r, \acute{s}) = \max Overlap(r, \acute{s}_j)$ ,  $\acute{s}_j \in sub(s)$ 
15:      if  $Overlap(r, \acute{s}) = 0 \forall \acute{s} \in sub(s)$  then
16:        put  $r$  in  $g_k$ 
17:      end if
18:    end for
19:  end for
20:  if  $|\bigcup_{i=0}^{|S|} sub(s_i)| < 4$  then
21:    retrieve  $sup(s)$ 
22:    for all  $\acute{s} \in sup(s)$  do
23:      retrieve  $\acute{R} = \bigcup_{t \in syn(\acute{s})} res(t)$ 
24:      create group  $\acute{g}$ 
25:       $\forall \acute{r} \in \acute{R}$ , put  $\acute{r} \in \acute{g}$ 
26:    end for
27:  end if
28: end for

```

6.2.2 Experiments

We implemented Alg.1 into two web interfaces to perform KS-based search on the structures acquired from WordNet (**System 2, S2**) and ontologies (**System 3, S3**). We also developed a web interface to simulate the cluster-based presentation of results as provided currently by folksonomies, **System 1 (S1)**². To do that, the clusters of the query keyword are retrieved using the folksonomy API. All the resources tagged with the

²<http://www.flickr.com/photos/tags/TAG/clusters/>

keyword are divided into groups according to the number of tags they share with each cluster.

All systems display the results grouped in meaningfully named groups. Figure 6.2 shows a screenshot with results from S1 for the query *sport*. For each group, there is a descriptive header which contains the title and the number of results per group. For S1 the title consists of the three most popular tags of the cluster (in accordance to the folksonomy clustering paradigm). For S2 and S3 the titles consist of the synonyms of the sense under which the results are clustered. For example, for this query, S2 returned groups described as *track and field*, *skiing* or *judo*, while S3 had groups named *golf*, *hunting* and *horseback riding*. The “see all” link allows the user to view all the results of the group when there are more than five results per group.

As a basis for our experiments, we used the MIRFLICKR-25000 [HL08] dataset proposed for ImageCLEF 2009. This contains 25000 images from Flickr with 69099 distinct tags. Although this is a dataset proposed for image analysis and 9% of the images are not tagged, the rest are tagged with a number of tags ranging from one to 75, spanning various domains.

We conducted three experiments. First, we enriched the dataset with strategies A and B (Sec.6.2.1) and evaluated the enrichment in terms of quantitative (Sec.6.2.1: (I), (II), (III), (IV), (VI)) and qualitative measures (Sec.6.2.2). Second, we performed a user evaluation on search (Sec.6.2.2). Finally, we used the user queries to measure $|\overline{inc}(T)|$ (V) (Sec.6.2.2).

Step 1. Enrichment Evaluation

The values obtained for metrics (I) to (IV) and (VI), defined for the evaluation of the enrichment, are shown in Tbl.6.1. In terms of the tagset coverage of the two knowledge sources, we observe that WordNet covers more tags than online ontologies (26.3% of all distinct tags, vs. 16%). We also note that a larger amount of tags were exclusively enriched by WordNet (12%) than ontologies (2.3%). Indeed, WordNet was better than ontologies in enriching concrete instances (*Hannover*) which was a limitation of Strategy B, as well as tags in a plural form. In a complementary fashion, ontologies outperformed WordNet by enriching non-noun tags (*alpine*), which was a limitation of Strategy A, or non-English tags (*collezione*).

One of the reasons for the low coverage by both sources is that, 71.4% of the tags were not mapped to any of the KS. This was due to phenomena such as compound tag concatenation (*rowingboats*), misspellings (*rasberry*), non English tags (*chaminá*), idiosyncratic tags (:D), tags that are not defined in either source (*augor*) and phrases (*daughters of the american revolution*).

Additionally, the major difference in coverage between WordNet and ontologies can be (at least partially) explained by the difficulty of matching tags to the concepts of these sources. Indeed, while the matching mechanisms of WordNet support the mapping of plurals and lexical variations to correct senses, matching is more difficult in ontologies. We found, for example, that ontologies use different **modelling styles** to express the names of entities, using one or more of the following mechanisms: the local name, rdf:label, rdf:comment or even locally specified properties (e.g., *O2::name =fiat*). In addition, the delimitation of compound labels (e.g., *O1#zantedeschia__genus_zantedeschia,O2#FloweringPlant*) is inconsistent across ontologies.

In terms of the richness of the created structures, WordNet provides, on average, more senses per tag (2.9) than ontologies (1.8). The amount of synonyms per senses is comparable in both sources, but important differences can be observed in the average number of more generic and more specific senses created in the two structures. Indeed, the structure created with WordNet, has a higher number of subsenses (2.7) on average than ontologies (1.5). Inversely, ontologies lead to more supersenses (1.5) than WordNet (1.0). One explanation for this is that online ontologies often express different points of views, or cover more domains than WordNet does and therefore lead to more supersenses. For example, *mosquito* has three more generic senses in ontologies and only one in WordNet:

ontologies: *Mosquito* $\xrightarrow{subSense}$ {*BloodFeedingArthropod*, *Insect*, *Vermin*}

WordNet: *Mosquito* $\xrightarrow{subSense}$ *Dipterous Insect* $\xrightarrow{subSense}$ *Insect*

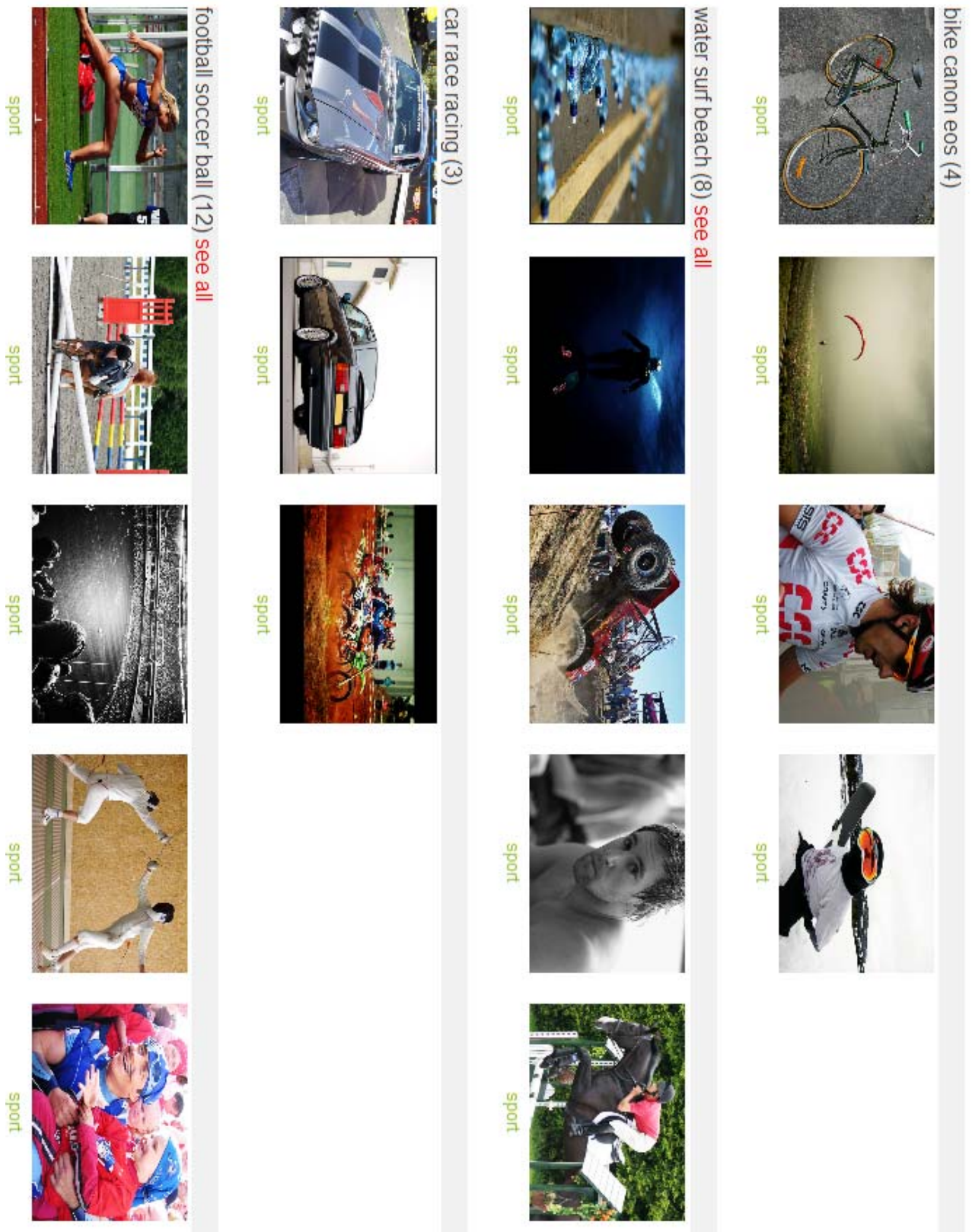


Figure 6.2: Result screenshot for the query *sport* in system S1.

	Measure	WordNet	Ontologies
(I)	$ \overline{syn}(S) $	2.3	2.2
(II)	$ \overline{senses}(T) $	2.9	1.8
(III)	$ \overline{sub}(S) $	2.7	1.5
(IV)	$ \overline{sup}(S) $	1.0	1.5
(V)	$ \overline{inc}(T) $	38%	39%
(VI)	$ \overline{T}_{KS} $	26.3%	16%

Table 6.1: Quantitative results of the enrichment evaluation

We also observe **variable hierarchical granularity** between ontologies and WordNet. Indeed, as shown for *mosquito*, the WordNet definitions of terms tend to be more fine-grained than in ontologies. Additionally, differences in the granularity of the definitions can also be observed within WordNet itself. For example:

WordNet: *Orange* $\xrightarrow{subSense}$ *Citrus* $\xrightarrow{subSense}$ *Edible Fruit*

WordNet: *Apple* $\xrightarrow{subSense}$ *Edible Fruit*

Step 2. User-based Search Evaluation

In the second experiment we performed a user study. The user group consisted of 25 expert and non expert users with basic knowledge of image search on the web. Their task was to post at least three single keyword queries to systems S1, S2 and S3 without domain or any other restrictions. We limited the search to single keywords because we were interested in comparing the richness of the structures created in Step 1 per keyword. In addition we maintained the same terms to compare with S1, which simulates the cluster-based search which is only available for single keywords. We obtained 88 distinct queries and the evaluators had to report on their comparative experience on using S1, S2 and S3. More specifically, they had to report on the questions of Tbl. 6.2. In Q2 and Q3 they had to select from a scale of 1 to 4; 1 being very unhelpful/all incorrect and 4 very helpful/all correct. They also had to report which results were most (ir)relevant/(in)correct and why for each query and system.

Question	S1	S2	S3
Q1: Did you find what you were looking for?	90%	85%	84%
Q2: How helpful was the presentation of the results and why?	2,9	2,8	2,8
Q3: Rate the number of correct versus incorrect results	3,3	2,8	3,1
Q4: Which is the best performing system?	35%	32%	33%

Table 6.2: User Questions and Responses

Overall, S1 performs better than S3 which performs better than S2 as seen in Tbl.6.2. Considering that in S2 and S3 none of the results of S1 are excluded (Alg.1), a possible explanation for this result is the reported decrease in precision (Tbl.6.2, Q3). The users stated that **S1 performed better because there were less groups** and it was easier to navigate through the results. It should be stressed that all the results returned from S1 were tagged with the query keyword. S2 and S3 included results tagged with tags related to the keyword, thus increasing the number of groups. In some cases the users reported that the results of some of these additional groups were irrelevant.

The effect of irrelevant results was maximised by an additional factor. In some cases the users reported that the **photos were tagged incorrectly**. This can be further justified from the result of Q3:S1 = 3.3/4 (Tbl.6.2). An example of this is the query *tiger*. Among the groups containing photos of tigers, S1 returned a group headed with $\{butterfly, shallowtail\}$ containing one image of a tiger butterfly. This was reported as incorrect because the user was unaware of this sense of the word *tiger* and no further explanation was given from the

system. This is a common phenomenon arising from categorising photos based on clusters of tags derived from co-occurrence. The **relations among the tags are not clear** (e.g., *tiger is a type of butterfly*) and it is not possible to give a justification for the retrieval of results and their categorisation in a particular group. This, however, would be possible if the knowledge $Tiger\ Butterfly \xrightarrow{subSense} Butterfly$ was provided by a KS.

In some cases the users reported that the presentation of S2 and S3 was more helpful even when the results returned by S1 were almost the same. According to them, the **images were presented under a meaningful category**. For example, for the query *horse*, S2 and S3 returned different groups for *colt*, *palomino* as opposed to the groups returned by S1 *italy*, *cavallo*, *england*. They found this distinction of results helpful for understanding the kind of horse depicted.

In cases that the query keyword did not return meaningful results in S1, the users reported that S2 and S3 returned **more and a higher variety of results**. For example, querying for *soap*, most S1's results depicted bubbles but S3 returned results depicting shampoo because $Shampoo \xrightarrow{subSense} Soap$ was found in online ontologies. Equally, for *doggy* S1 retrieved only two images while S2 retrieved all images tagged with *dog* because *doggy* is one of the synonym terms for the sense of *dog* in WordNet. Finally the users were asked to select the system that performed better in all their queries (Q4). 35% of the users selected S1, 33% S3 and 32% selected S2. The responses to the rest of the questions of Tbl.6.2 justify this too. S1 performed better due to less groups of results. S2 and S3 returned better group descriptions and in addition S3 groups were judged to be more relevant.

Step 3. Quantitative Search Evaluation

Taking into account the user's queries and comments, we measured the approximate average $inc(T)$ for WordNet and ontologies. In Figure 6.3 WordNet's increase is represented with dark lines and ontologies' increase with light lines. $inc(T)$ is the ratio of additional correct results returned by the expansion of the query keyword divided by the total results returned as described in Sec.6.2.1 (V). Figure 6.3 demonstrates the $inc(T)$ for the user queries for which S2 and S3 returned additional correct results. The rest of the queries were either not mapped to any KS or their related senses did not map to the tagset.

$inc(T)$ is affected by two factors as shown in Sec.6.2.1 (V). The first, depends on the tagset and is not relevant to the KS used. This is **how popular is the search keyword k_i in the tagset**, i.e., how many resources are tagged with it (e.g., *soap*, *doggy*). The second factor, is the number of **correct additional resources** retrieved when expanding k_i with synonyms, subsenses and supersenses. This depends on the number of synonyms, sub/super-senses and how well these are covered in the tagset. A representative case of related senses that were not well covered is the one of *nature*. S2 returned four groups, one for each of the subsenses of *nature* in WordNet, i.e., $\{animality, complexion, disposition, sociality\}$. The users reported that the results of S2 were not significantly more than S1, they were generic and quite irrelevant to nature. On the other hand, S3 returned five groups for $\{sky, fire, mountain, reef, rice\}$ which are ontological subclasses of *nature*. All the above were well covered by the tagset and, with the exception of *rice*, the users reported that results and their grouping were meaningful and satisfactory. As a result the increase of *nature* from S3 was significant. The phenomenon of irrelevant groups was more frequent in S2 than S3, justifying the lower increase for S2 (Tbl.6.1) and the lower user satisfaction with this particular system (Tbl.6.2).

In cases of querying for *apple*, system S3 returned one group representing the sense of fruit. S2 returned two groups for apple derived from the two senses of apple in WordNet, i.e., *fruit* and *fruit tree*. However, none of the two senses was relevant to the sense of *computer company*. This shows that the number of senses is not enough information to decide if a KS can deal with polysemy. To decide this information on the coverage of the senses from the tagspace is required.

Another useful outcome emerged with querying for *may*. While S2 and S3 did not present the results in any meaningful manner nor did they return any additional results, S1 performed quite satisfactorily. Four clusters were returned grouping together images tagged with $\{england, london\}$, $\{spring, flowers\}$, $\{sky, cloud\}$ and $\{paris, france\}$. A plethora of photos shot in and tagged with *may*, can depict flowers, sky, cities and so on but may depict nothing that can symbolise the month May. This is a type of idiosyncratic tagging and

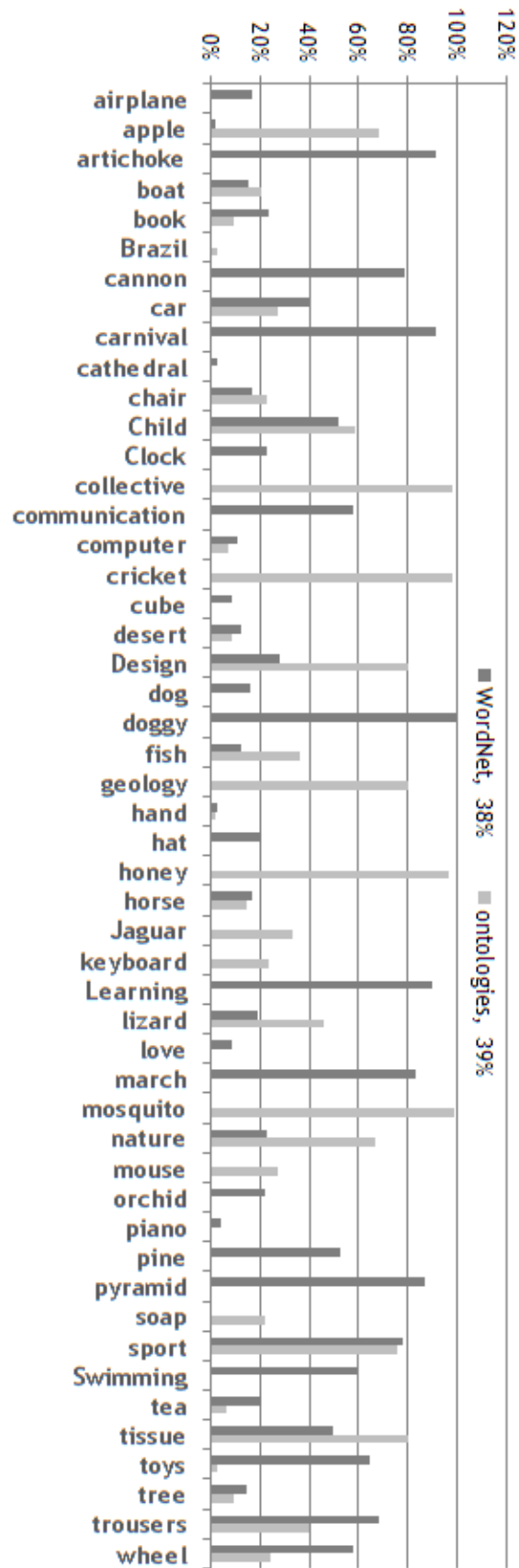


Figure 6.3: Increase in results on the user entered keywords

no KS can supply formal relations between *may* and these tags since there are no formal relations among them. Nevertheless, for this type of **idiosyncratic tagging the clustering of results based on frequent tag co-occurrence is quite efficient.**

6.2.3 Conclusions

In terms of tag-space enrichment, WordNet outperformed ontologies in most of the measures. It provided more senses per tag and more synonyms per sense than ontologies which is an indication of better addressing **tag polysemy** and **tag synonymy** respectively. Additionally, WordNet covered a higher percentage of tags than ontologies. WordNet and ontologies returned comparable measures for subsenses and super-senses which served as an indication for deciding which KS can project the **basic level variation** of the tag-space to a semantic structure.

However, in the user evaluation the ontologically created structure performed better in search than the WordNet created structure. The expansion of the query keyword with terms from ontologies returned a higher number of results compared to the expansion provided from WordNet despite the fact that WordNet provided a richer structure. This indicates that the tagset coverage of the created semantic structure is more important in the context of search than the richness of the structure. In other words, the *subSense* or *superSense* of the sense of a tag should be mapped well to the rest of the tags in the tag-space.

The conclusions from this experiment were exploited to define the second version of the FLOR enrichment algorithm, described in the next Section 6.3

6.3 Second Version of the FLOR Folksonomy Enrichment Algorithm

The updated version of FLOR enrichment algorithm was developed with the aim to eliminate the issues caused by the previous version of FLOR([VTAGS⁺08]) such as low enrichment percentage and low tag-space coverage. In addition, compared to the last version of FLOR, it implements a method for the integration of all the enriched tag-space into a unified semantic layer, SL. For example in Figure 6.4 we see the output of FLOR for **Resource_X**. The tags of the resource have been matched to semantic definitions, the senses, which in turn relate (A) with the senses of the tags in the same tagset e.g.,

$$X_Balaton \xrightarrow{\text{hasDefinition}} Balaton \xrightarrow{\text{partOf}} Hungary \xleftarrow{\text{hasDefinition}} X_Hungary$$

and (B) with the senses of tags of other tagsets, e.g., *Resource_...*. For example:

$$X_Balaton \xrightarrow{\text{hasDefinition}} Balaton \xrightarrow{\text{typeOf}} Lake \xleftarrow{\text{hasDefinition}} \dots_Lake$$

The output of FLOR, an example of which is depicted in Figure 6.4, populates the FLOR ontology described in Section 6.3.2

6.3.1 FLOR Algorithm

FLOR algorithm operates in three main phases as depicted in Figure 6.5. Phase 1, the **Lexical Processing** (See Sec.6.3.3) deals with the lexical issues that affect the enrichment. Phase 2, the **Sense Selection** (See Sec.6.3.4) is responsible for identifying the appropriate senses for the tags and incorporates methods such as disambiguation and filtering of Semantic Entities. Finally the last, Phase 3, **Semantic Aggregation** (See Sec.6.3.5) integrates the senses discovered in Phase 2 into a connected SL that represents the tag-space.

The FLOR enrichment algorithm is presented in Alg. 2. For each tagset TS of a given tag-space (set of tagsets), phases 1 and 2 are executed. The output of Phase 2 for each tagset is a number of senses representing the meaning of its tags. The senses for all enriched tagsets are maintained in a sense repository.

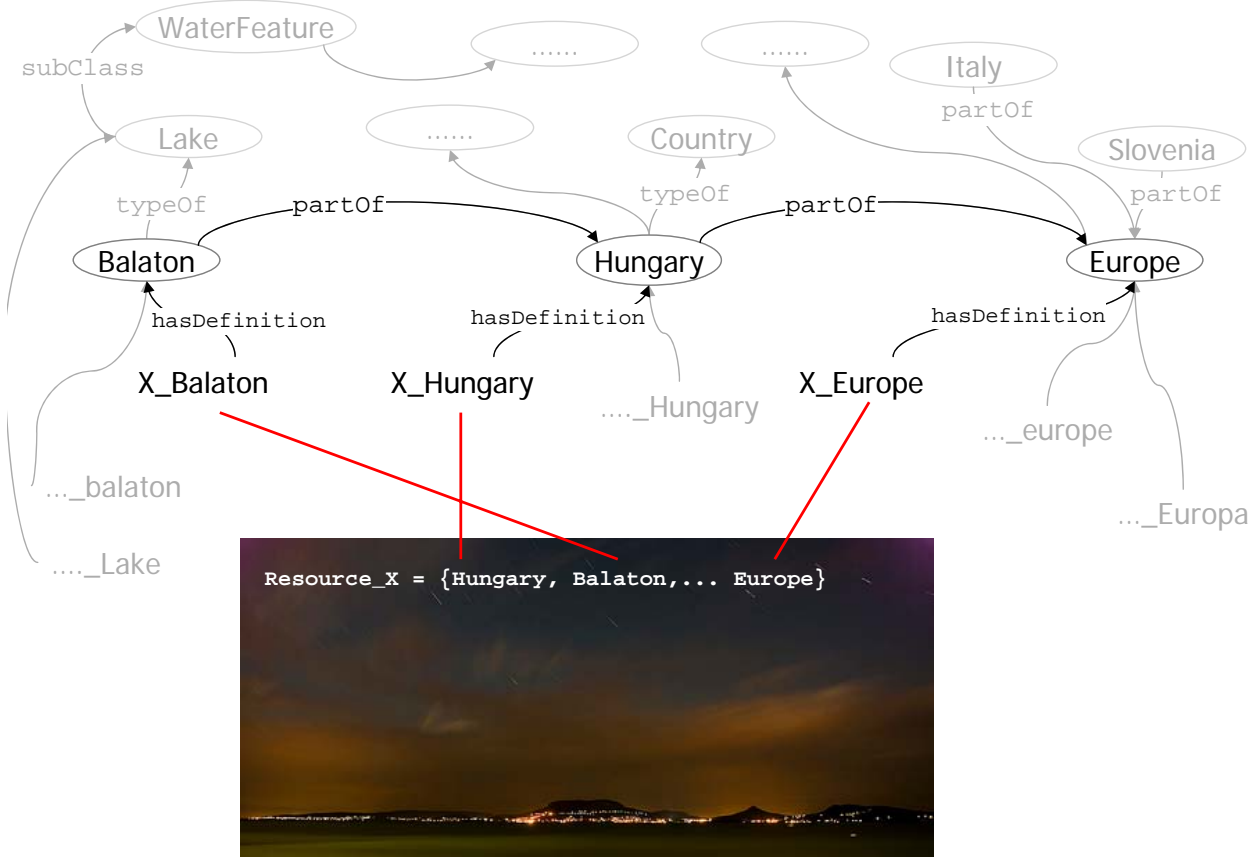


Figure 6.4: Example of FLOR enrichment for the tagset of Resource_X

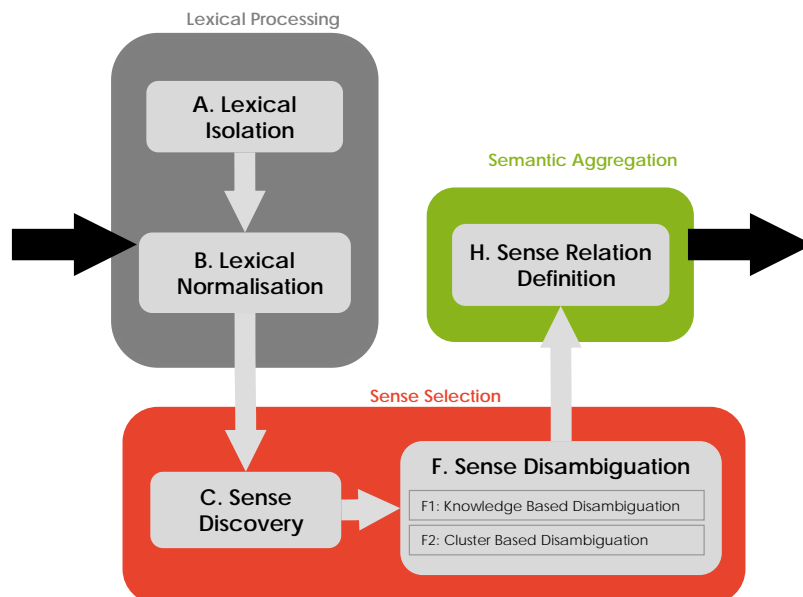


Figure 6.5: The updated version of FLOR Enrichment Algorithm

Once all tagsets have passed the lexical processing and sense selection phases the sense repository is processed by the semantic aggregation phase which in turn produces the output Semantic Layer, SL. The algorithms for Lexical Processing, Sense Selection and Semantic Aggregation are explained in more detail in sections 6.3.3, 6.3.4 and 6.3.5 respectively.

Algorithm 2 The updated version of FLOR Enrichment Algorithm

```

1: Create SenseRepository
2: for all tagset  $TS \in \text{tagspace}$  do
3:   LexicallyProcess(TS) ▷ Phase 1, see Alg. 3
4:   Update(SenseRepository, SelectSenses(TS)) ▷ Phase 2, see Alg. 4
5: end for
6: SemanticLayer = SemanticallyAggregate(SenseRepository) ▷ Phase 3, see Alg. 7
   return SemanticLayer

```

6.3.2 FLOR Ontology

Before proceeding to the details of the algorithms of each phase, it is important to explain the ontology on which the expected structure (see example in Figure 6.4) is based. The FLOR ontology is depicted in Figure 6.6. As demonstrated in the figure this ontology consists of three classes and three properties.

Class: TaggedResource represents the folksonomy resources that are tagged with at least one tag. This class is the first part of the definition of the folksonomy relation that “*a tagged resource is tagged with tag*” and is introduced in Definition 1.

Class: SpecificTag represents the entities of tags. It should be pointed out that this definition refers to the specific tag that is assigned to a single resource. For example, in Figure 6.4 there are more than one tags with the name “europe”, X_Europe and \dots_europe . These two share the same label but, in fact, are instantiating the concept of tag in the two different resources. The SpecificTag class is the second part of “*a tagged resource is tagged with tag*”.

Property: isTaggedWith is the predicate in the relation of TaggedResource and SpecificTag. It represents the assignment of a Tag to a Resource, otherwise the fact that a resource is tagged with a specific tag. The domain of the property is the TaggedResource and the range is the SpecificTag. The cardinality of this relation is One to Many, i.e., one resource can be tagged with many tags, but each of these tags is only assigned to this resource.

Class: Sense formally represents the concept, sense or meaning that is assigned to a tag. This class conceptualises all the definitions assigned to the tags by FLOR. More details on this class are given in Sec. 6.3.5.

Property: hasDefinition This property models the relation among the specific tag in a tagset and the Sense. Its domain is the SpecificTag class and the range is the Sense and the cardinality is Many to One, i.e. many specific tags may have the same sense, but the meaning of the tag in the context of the particular TaggedResource is uniquely assigned to one sense. This property, along with the Sense class, models the output of FLOR. The other three entities in this ontology are representing the relations of entities in folksonomies as they are described in Definition 1.

Property: isFoundIn this is a property that was created to represent the fact that a sense is created by aggregating semantic entities, SEs, located in various KS (details in Section 6.3.4). The relation isFoundIn should be One to Many as one sense is unambiguously defined by all the possible ontological entities that exist in the semantic web and these only define the particular sense. However this is not always the case. The SE can be any ontological URI that has contributed to the creation of this Sense

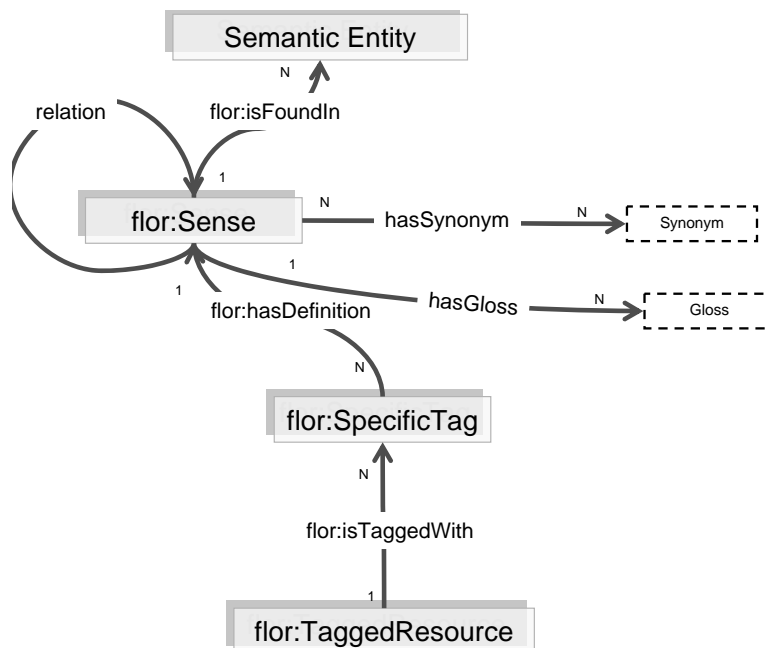


Figure 6.6: Ontology of FLOR

class or the WordNet synset that contributed to the definition.³ However this is not always possible and there are cases where the same URI may be found to define more than one sense.

Finally there is an abstract entity called “relation”. The reason why it is abstract is because it can be either of the following:

flor:subSense is used to represent more specific senses of a sense. This is a looser relation than the `rdfs:subClassOf` because apart from subclasses it represents individuals and WordNet hyponyms.

flor:superSense equally with its inverse, `flor:subSense`, this relation represents a union of the inverse of `rdfs:subClassOf`, the hypernyms and the classes of instances.

flor:hasPart is a container for all the ontological relations for meronymy as well as all the types of meronyms in WordNet such as substance meronyms, part meronyms and member meronyms.

flor:isPartOf is the inverse relation of `flor:hasPart`.

6.3.3 Lexical Processing of Tags

The goal of the Lexical Processing phase is to (A) isolate tags that cannot be further enriched and slow down the overall process and (B) to lexically enrich the tags so that we can achieve higher coverage of tags in the respective KS.

A. Lexical Isolation. We have used a variety of heuristics to identify which tags are the ones to be isolated. For each type of idiosyncratic tags we implement an isolator function.

³In the WordNet case the URI is created of the FLOR namespace and the WordNet synset ID created by the offset of the synset and its part of speech. This was done as an easy way to dereference the synset.

Algorithm 3 LexicallyProcess(TS): Lexical Processing of tagset TS

```

1: for all  $Tag\ t \in TS$  do
2:   for all  $Isolator$  do                                     ▷ See. Tbl 6.3 for different types of isolators selected
3:     if  $Isolator(t)$  TRUE then
4:       remove  $t$  from  $TS$ 
5:     end if
6:   end for
7: end for

```

In the following table we describe the different types of Lexical Isolators we developed for each type of tags we wish to isolate. Depending on the configuration of the flor process and the nature of the tagspace to be enriched, a different selection of isolators can be made. For example if the tagspace does not contain noisy tags, then the selection of a Noisy Tags isolator is not required. Or, if we don't want to isolate the infrequent tags then the Infrequent Tags isolator can be excluded.

Isolator	Purpose
Infrequent Tags	Tags that do not appear frequently enough in the tagspace
Idiosyncratic Tags	Tags that make sense only to a user or to a group of users. They are hard to define and are usually quite frequent. For example, <i>macro</i> , <i>color</i> , <i>flickrdiamond</i> are some examples of idiosyncratic tags
Noisy Tags	Tags with special characters, exif data, combinations of letters, symbols and numbers, e.g., <i>d80</i> , <i>135mm</i> , <i>:D</i>

Table 6.3: Different Types of Tag Isolators

B. Lexical Normalisation is essential in order to achieve a better anchoring of tags to the various KS. For example in a tagset:*{flower, rose, flowers, garden, spring}* the *flower* and *flowers* will be considered the two different labels of the same tag and the entities discovered for both these labels of them will be assigned to one. In the previous version of FLOR, different senses would be assigned to *flowers* and *flower* due to the different label description. In addition, in cases of tags such as *santaBarbara*, the different delimitations such as *santa barbara*, *santa-barbara* will also be included in the tagspace.

6.3.4 Sense Selection for Tags

This phase is the most significant part of the FLOR algorithm. Here, the meaning of the tags is decided, the senses are created to represent this meaning and they are linked to the tags. In the following sections we analyse in detail the steps of the Sense Selection phase.

Algorithm 4 SelectSenses(TS): Sense Selection for the tags of the TS

```

1: for all  $Tag\ t \in TS$  do
2:    $DiscoverSenses(t)$                                      ▷ see Alg. 5
3:    $DisambiguateSenses(t, TS)$                              ▷ see Alg. 6
4: end for
   return  $ListOfSenses(TS)$ 

```

Sense Discovery

The sense discovery phase is responsible for the anchoring of tags to semantic entities.

Algorithm 5 DiscoverSenses(t): Discover the candidate senses for tag t

-
- 1: *AnchorToEntities(t)*
 - 2: *ClusterSenses(t)*
-

Intuitively, for each label L of a tag (usually $|L| = 1$ but more may be produced by the lexical normalisation phase) the sense discovery algorithm tries to locate semantic entities that have L as part of their lexical description.

Matching Tags to ontologies via Watson Following the NeOn paradigm of re-using ontologies, this algorithm exploits all the existing ontologies that are indexed in Watson. Utilising the Watson API, the algorithm queries Watson for semantic entities, namely Classes and Individuals, that contain the label(s) of a tag either in their localname (ID) or in the property `rdf:label`. The anchoring of the tag labels to such SEs is initially performed strictly, however in case no entities are returned the constraints are relaxed and the also SEs whose lexical description flexibly matches the label of the tag are also returned.

Matching Tags to WordNet The matching of tag labels to WordNet is also done using the WordNet api and again, in case there are WordNet synsets strictly matched against the tag, there is no further attempt for flexible matching. Finally it should be pointed out that the Semantic Entities in the case of WordNet are considered the WordNet synsets, each of which describes a (usually) distinct meaning. In addition, only noun and adjective synsets are targeted. Verb and adverbs are excluded.

The output of the Sense Discovery step is a list of candidate senses obeying to the Sense structure depicted in Figure 6.6. Each semantic entity, either returned from Watson or from WordNet, is transformed to a sense. For example in the figure 6.7 we see the description of a Semantic Entity returned from Watson when querying for *Lake*: <http://a.com/ontology#Lake>.

This entity is transformed to a sense, e.g., *sense_lake_1*. The localname of the SE URI is added to synonym list of the sense, the comment is added to the gloss list of the sense, the URI itself is added in the list of entities the sense is found in. In case the SE contains labels, these are also added into the synonyms list. Finally the relations of the SE with other SEs are appended to the sense in the following way. For example the relation:

$$\text{http://a.com/ontology\#Lake} \xrightarrow{\text{rdfs:subClassOf}} \text{http://a.com/ontology\#WaterBasin}$$

is transformed to

$$\text{flor:sense_lake_1} \xrightarrow{\text{flor:subSense}} \text{http://a.com/ontology\#WaterBasin}$$

according to the assumption (described in section 6.3.2) that all relations of the type `rdf:subClass` are transformed to the `flor:subSense` which conveys the narrower concept meaning but in a less specific way. Although in Figure 6.6 we can see that the relations added to the sense have as domain and range the class of Sense, the process of transforming the entity <http://a.com/ontology#WaterBasin> into a sense is taken care of by the last phase of FLOR, Semantic Aggregation.

By using multiple ontologies and WordNet, the same concept may be defined more than once thus leading to different types of redundancies such as:

1. Redundancy of the same entity. Several ontologies declare the same URI.
2. One entity with the same id is declared in two different versions of the same ontology, e.g., *O1.daml:plant* and *O1.owl:plant*.
3. The same concept is declared in different ontologies in the same manner, namely it is subsumed by the same concept(s) and has the same ontological neighborhood (relations, literals and so on) but different URI.

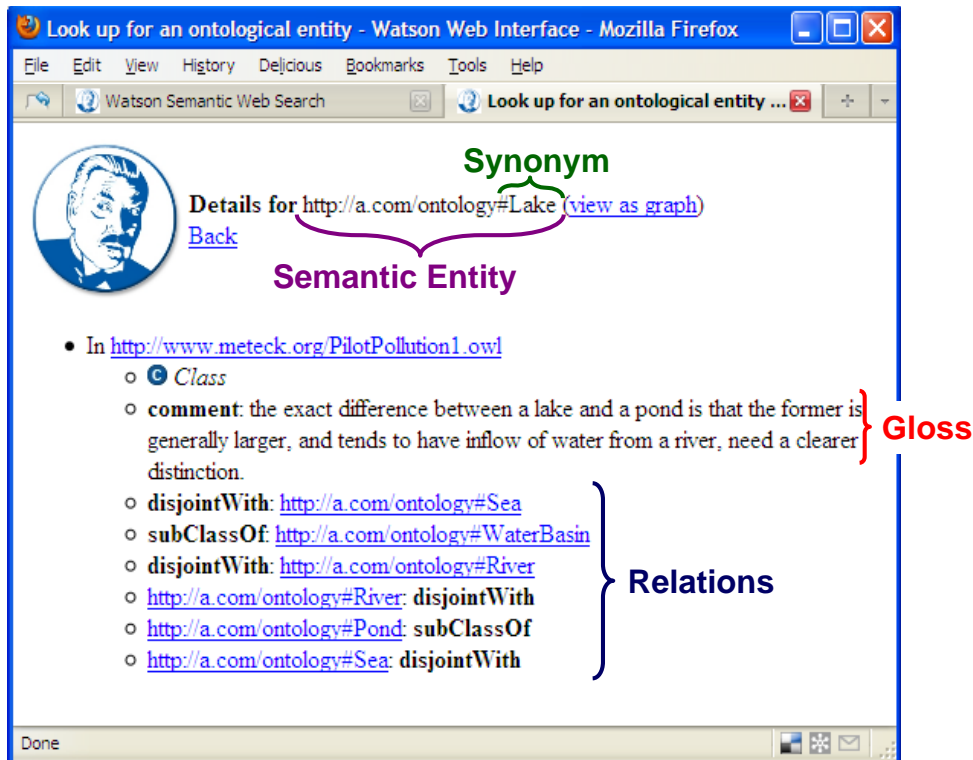


Figure 6.7: An example Semantic Entity returned from Watson when querying for *Lake*

4. The same concept is defined in different ontologies by two different entities with different neighborhood, e.g., $O1:Banana \xrightarrow{subClassOf} \{O1:GroceryProduce, O1:TropicalFruit\}$ and $O2:Banana \xrightarrow{subClassOf} O2:Tree-Fruit$

The **clustering algorithm** minimises these redundancies by grouping sufficiently similar semantic descriptions of senses together and merging them into a new description, a cluster of senses, which we consider one sense. The algorithm is repeated until all obtained senses are sufficiently different from each other. To compute the similarity between two entities we compare their semantic neighborhoods (superclasses, subclasses, disjoint and equivalent classes and named relations) as well as their lexical information (localnames, labels). The similarity $Sim(e_1, e_2)$ for two entities e_1 and e_2 is computed as:

$$Sim(e_1, e_2) = W_L \times Sim_L(e_1, e_2) + W_G \times Sim_G(e_1, e_2)$$

$Sim_L(e_1, e_2)$ is the similarity of the lexical information of the two entities computed using the Levenshtein metric. $Sim_G(e_1, e_2)$ is the similarity of the entities' neighborhood graphs. For example, the superclasses of e_1 are compared against the superclasses of e_2 . This is repeated for all the neighbor entities of e_1 and e_2 . The similarity among the neighbor entities is computed based on string similarity too. For this experiment we set a low similarity threshold of 0.3 in order to achieve a maximum clustering result. In addition, we set $W_G = W_L = 0.5$ in order to cater for the heterogeneity of online ontologies in terms of the richness of their lexical and structural information. For example, for *banana* we obtained a single cluster because, according to our clustering algorithm, there is only one sense of banana in all online ontologies. This cluster of senses contributes to the sense of *banana* with synonyms derived from the localnames and labels

$$\{L_1: "banana", L_2: "an elongated yellowish fruit which grows on palm trees"\}$$

and leads to the structural information

$$\text{Banana} \xrightarrow{\text{subClassOf}} \{\text{Fruit}, \text{Tropical Fruit}, \text{GroceryProduce}, \text{Tree Fruit}\}.$$

L_2 was the label of one of the clustered senses. ⁴.

The cluster of senses is represented as a single sense, following the representation of sense in Figure 6.6. All the information of the senses that contributed to the creation of this clustered sense is merged and appended into the respective property of the clustered sense. For example, one of the most common differences among one sense that has been created from an ontological entity, and one sense that is the result of the clustering of senses is the number of “*flor:isFoundIn*” entities.

After the clustering of senses the Sense Discovery step of the Sense Selection phase returns a number of candidate senses for each tag. Usually, after the clustering algorithm, and because significantly similar senses have been merged into a clustered sense, there are up to three candidate senses for each tag. These are the input of the Sense Disambiguation step which in turn decides which of these candidate senses is the most appropriate to describe the meaning of the tag in the context of this tagset.

Sense Disambiguation

The Sense Disambiguation step is responsible for assigning at most one sense to each tag. Only the tags that have been assigned one or more senses are subject to this phase. Even when the previous step, Sense Discovery, has returned only one sense for a tag, the Sense Disambiguation step is responsible to validate whether the meaning of this sense is correct or not in the context of this tagset.

The disambiguation algorithm first performs a relation based disambiguation and if this step fails it performs a statistical disambiguation for t . The two methods are described below.

Algorithm 6 DisambiguateSenses(t, TS): Disambiguation of t 's candidate senses based on the context of TS

```

1: Create ListOfSenses
2: for all Sense  $s \in t.Senses$  do
3:   if RelationBasedDisambiguation( $s, TS$ ) == TRUE then
4:     put  $s$  in ListOfSenses
5:   else
6:     if StatisticalDisambiguation( $s, TS$ ) == TRUE then
7:       put  $s$  in ListOfSenses
8:     end if
9:   end if
10: end for
    return ListOfSenses

```

Relation Based Disambiguation The relation based disambiguation exploits the relations among the candidate senses of the tags. Intuitively, the candidate senses of a tag that relate with the candidate senses of most of the rest of the tags are selected. This means that only the tags that have been assigned candidate senses from the sense selection phase are exploited in this type of disambiguation.

For each candidate sense of each tag, all the entities that relate with it are selected. For each of these entities, their semantic neighbourhood is extracted. The semantic neighborhood of an entity consists of the entities that are related with it in the ontology (or in WordNet) with paths of length up to 4. For example the following path belongs to the semantic neighborhood of <http://ontosem.org/#apple>.

$$\begin{array}{l} \text{http} : // \text{ontosem.org} / \# \text{apple} \\ \xrightarrow{\text{subClassOf}} \text{http} : // \text{ontosem.org} / \# \text{tree} - \text{fruit} \end{array}$$

⁴Different ontologists have different representation styles and may include a comment as a label. We avoid the usage of a long description (formerly used as label) to be assigned to their synonyms as opposed to the glosses of the sense we employ a variety of heuristics.

$$\begin{array}{l}
 \xrightarrow{\text{subClassOf}} \text{http://ontosem.org/\#fruit - foodstuff} \\
 \xrightarrow{\text{subClassOf}} \text{http://ontosem.org/\#plant - derived - foodstuff} \\
 \xrightarrow{\text{subClassOf}} \text{http://ontosem.org/\#foodstuff}
 \end{array}$$

The candidate senses are selected with the following criterion. The sense whose semantic neighborhood contains the most of the related entities of the other senses of the tagset is selected.

This disambiguation is using relation discovery based on a single resource. For example, the hierarchical relation *rdfs:subClassOf* among *Ontology1:Fruit* and *Ontology2:Apple* will not be retrieved. To overcome this problem plan to integrate with FLOR the upcoming functionality of cross ontology relation discovery from SCARLET.

Statistical Disambiguation The statistical disambiguation was added in the second version of the FLOR algorithm to cater for cases that the relation based disambiguation would fail. The previous experience with FLOR enrichment showed that this is a frequent case since the tags within a tagset relate with each other with generic, rather than hierarchical, relations. The same time the majority of relations found either in online ontologies or in WordNet are hierarchical. As a result the relation based disambiguation was successful for a low percentage of the cases.

The statistical based disambiguation algorithm exploits the synonyms of the sense as well as its relations. In addition, it uses the tags of the tagset that were not assigned any candidate senses from the sense selection phase. Intuitively, for each candidate sense of tag *t* it creates a vector of related terms derived from the synonyms of the sense and the labels and localnames of the entities which these sense are related. For example for the sense depicted in Figure 6.7 the vector of related terms would be: $V = \{Lake, WaterBasin, Pond\}$ since these entities are related with the sense with any of the four relations described in Section 6.3.2. Consider an example tagset $TS = \{lake, fish, lawn\}$. The mean relatedness of V and TS is calculated for the related vector V_i of all the candidate senses for the tag lake. The sense whose vector has a higher mean relatedness with the tagset is selected.

The mean relatedness of each sense's related vector is calculated by the aggregated relatedness of each vector element with each of the tags of TS (apart from lake). For the relatedness among two terms we used a modified version of the Google Relatedness measure [CV07]. We used the statistically derived clusters of tags over a certain tagspace in the place of web documents.

6.3.5 Semantic Aggregation of Senses

The last phase of FLOR, Semantic Aggregation of Senses, is responsible for generating the semantic layer of senses and their relations similar to the one depicted in Figure 6.4. The output of the sense selection phase is a list of senses according to Figure 6.6 with the difference that the relations of the senses do not have as a value another sense but a Semantic Entity. For the example of the tag *Lake_X* in the *resource_X* (Figure 6.4) we have

$$X_Lake \xrightarrow{\text{hasDefinition}} Balaton \xrightarrow{\text{isFoundIn}} \text{http://www.ontotext.com/kim/2004/04kimo\#Lake}$$

The same output is also derived for the rest of the tags in resource *X* and all the rest of the resources of the tagspace. In this case there is no explicit relation among the senses of the tags. Such as:

$$Balaton \xrightarrow{\text{typeOf}} Lake$$

where both *Balaton* and *Lake* are senses.

The Semantic Aggregation phase takes as input all the individual senses and examines the Semantic Entities that these senses are related with. Intuitively, the semantic aggregation performs in the following steps. Selects all the entities that are related to a sense with the same relation, e.g. *subSense*. Considers these

entities in Figure 6.8 senses to be SE_m, SE_n and SE_q . For each of these entities retrieve the senses that have as value these entities in the “isFound” property. We see that only SE_m and SE_n are set to be as a value in the “isFound” property of the same sense, $Sense_2$. This provides some evidence that the relation that is connecting $Sense_1$ with these entities should be leveraged to relation among $Sense_1$ and $Sense_2$. The additional step is to look for potential inverse relations among the isFound entities of $Sense_1$ and the related entities of $Sense_2$. In this example we see, indeed, that there are SE_2 and SE_3 that are values of the isFound property of $Sense_1$ and are subSenses of $Sense_2$. This further strengthens the evidence that

$$Sense_1 \xrightarrow{subSense} Sense_2 \text{ and } Sense_2 \xrightarrow{superSense} Sense_1$$

For the entity SE_q that was not in the isFound value for $Sense_2$ there are two options. In case it is found in another sense, e.g., $Sense_3$ then following the same procedure as above we leverage the relation of

$$Sense_1 \xrightarrow{subSense} SE_q \xleftarrow{isFound} Sense_3 \text{ to } Sense_1 \xrightarrow{subSense} Sense_3$$

In case this is not found in any other sense we check the similarity of this entity to the other entities that connect to the sense with the same relation. In most of the cases the similarity is lower than our merging threshold and then we create a new sense using the information of this entity only and leverage the relation to this sense. In the unlikely case that the similarity is higher than a certain threshold then the related entity, is appended in the values of isFound to the sense with whose entities shares the higher similarities.

The reason why a sense for every related entity is not created is because we are not aware of all the possible entities this one could be clustered with in order to create a sense. In addition to this, if this related entity only appears once in the whole process, e.g. in the related entities of another sense, then the computational effort of searching for relevant entities and comparing them and clustering them in order to create another sense, can not be justified. Alternatively, if more than one related entities have been found in the enrichment process, then the comparison and integration of these entities would justify such an effort.

Algorithm 7 SemanticallyIntegrate(SenseRepository): Aggregates all the senses discovered by sense selection into the output semantic layer

```

1: for all  $Sense\ S \in SenseRepository$  do
2:    $SenseRepository = IntegrateSense(S)$ 
3:   for all  $Relation\ R \in Relations(S)$  do
4:      $SenseRepository = IntegrateRelation(R)$ 
5:   end for
6: end for return  $SenseRepository$ 

```

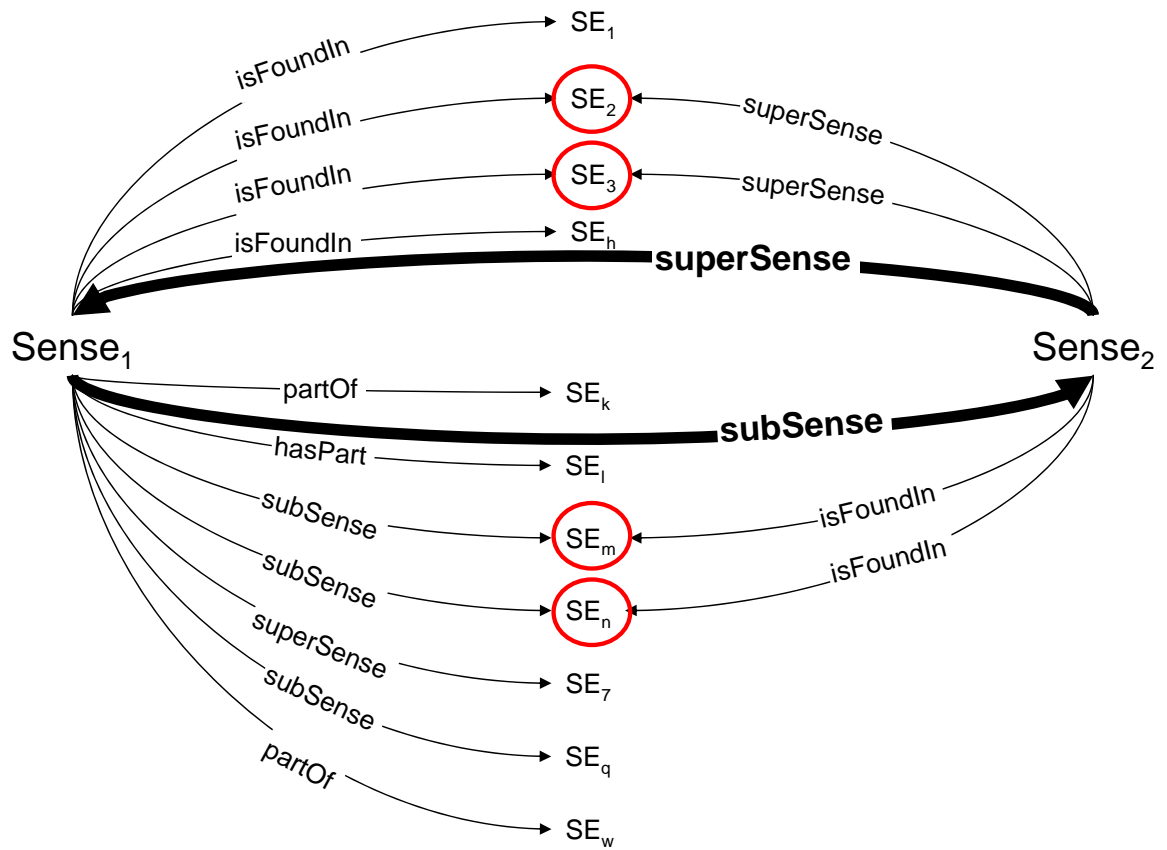


Figure 6.8: Two Senses Sharing Semantic Entities (SE) provides evidence on their relations

6.4 Ongoing Work

Preliminary evaluations on the second version of the FLOR algorithm give significant evidence on the improvement of the enrichment in terms of tag-space coverage and enrichment quality. However, there are a few open issues that we are currently working on.

Comparative Evaluation of the two versions of FLOR. In order to assess the improved version of the enrichment algorithm we are performing an expert based evaluation of the output generated by FLOR using the same tag-space we used to evaluate the previous version (reported in [VTAGS⁺08])

Task Based Evaluation of the enriched tag-space. In addition to the evaluation of the quality of the FLOR output we are interested in evaluating the usefulness of the semantic layer of FLOR in terms of a real task. For this reason we are performing a comparative user based evaluation on the task of search. The users have to report their experience on searching the tag-space using (A) the folksonomy-based search as currently provided by the existing folksonomies and (B) the intelligent search that is utilising the Semantic Layer returned by the FLOR enrichment algorithm.

Bibliography

- [ANS05] ANSI/NISO. *Documentation – Guidelines for the construction, format, and management of monolingual controlled vocabularies.*, 2005. Report ANSINISO Z3919.
- [ASM08] S. Angeletou, M. Sabou, and E. Motta. Semantically enriching folksonomies with FLOR. In *Proc of the 5th ESWC: CISWeb*, Tenerife, Spain, 2008.
- [BH06] S. Brockmans and P. Haase. A Metamodel and UML Profile for Networked Ontologies. A Complete Reference. Technical report, Universität Karlsruhe,, 2006.
- [Bra05] D. Brandon. Recursive database structures. *Journal of Computing Sciences in Colleges*, 2005.
- [Byr92] E.J. Byrne. A conceptual foundation for software re-engineering. In *International Conference on Software Maintenance and Reengineering*, pages 226-235. IEEE Computer Society. 1992.
- [CR09] O. Corcho and C. Roussey. Synonym or equivalence pattern. 2009.
- [CV07] R.L. Calibrasi and P.M. Vitanyi. The Google Similarity Distance. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):370 – 383, 2007.
- [dL09] Mathieu d’Aquin and Holger Lewen. Cupboard – a place to expose your ontologies to applications and the community. In Lora Aroyo, Paolo Traverso, Fabio Ciravegna, Philipp Cimiano, Tom Heath, Eero Hyvönen, Riichiro Mizoguchi, Eyal Oren, Marta Sabou, and Elena Paslaru Bontas Simperl, editors, *The Semantic Web: Research and Applications, 6th European Semantic Web Conference, ESWC 2009, Heraklion, Crete, Greece, May 31-June 4, 2009, Proceedings*, volume 5554 of *Lecture Notes in Computer Science*, pages 913–918. Springer, MAY 2009.
- [DR93] J. Dumas and J. Redish. A practical guide to usability testing. *Exeter, UK Intellect*, 1993.
- [ES07] Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007.
- [GH06] S. Golder and B. Huberman. Usage patterns of collaborative tagging systems. *Journal of Information Science*, 32:198–208, 2006.
- [GKRT04] Ramanathan V. Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. Propagation of trust and distrust. In Stuart I. Feldman, Mike Uretsky, Marc Najork, and Craig E. Wills, editors, *WWW*, pages 403–412. ACM, 2004.
- [Guh03] R. Guha. Open Rating Systems. Technical report, Stanford University, CA, USA, 2003.
- [HKTR04] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.
- [HL08] M. Huiskes and M. Lew. The mir flickr retrieval evaluation. In *Proc. of the ACM Int. Conf. on MIR*, 2008.

- [ISO98] editor. ISO 9241-11 ISO. Ergonomic requirements for office work with visual display terminals (vdt) Ú part 11: Guidance on usability, 1998.
- [ISO04] International Standard Organization (ISO). *Information technology - Metadata registries - Part 1: Framework*, 2004. Report ISO/IEC FDIS 11179-1.
- [KC93] J. Kirakowski and M. Corbett. Sumi: The software usability measurement inventory. *British Journal of Educational Technology*, 24(3):210–212, 1993.
- [Knu76] Donald E. Knuth. Big omicron and big omega and big theta. *SIGACT News*, 8(2):18–24, 1976.
- [LEC07] D. Laniado, D. Eynard, and M. Colombetti. Using WordNet to turn a folksonomy into a hierarchy of concepts. In *Proc. of 4th SWAP*, 2007.
- [LSNM06] H. Lewen, K. Supekar, N.F. Noy, and M.A. Musen. Topic-Specific Trust and Open Rating Systems: An Approach for Ontology Evaluation. In *Proc. of the 4th International Workshop on Evaluation of Ontologies for the Web (EON2006) at the 15th International World Wide Web Conference (WWW 2006)*, Edinburgh, UK, MAY 2006.
- [LY07] S. Lee and H. Yong. Tagplus: A retrieval system using synonym tag in folksonomy. In *Int. Conf. on Multimedia and Ubiquitous Engineering*, 2007.
- [MZ06] E. Malinowski and E. Zimányi. Hierarchies in a multidimensional model: From conceptual modeling to logical representation. *Data and Knowledge Engineering*, 2006.
- [SAd⁺07a] M. Sabou, S. Angeletou, M. d'Aquin, J. Barrasa, K. Dellschaft, A. Gangemi, J. Lehman, H. Lewen, D. Maynard, D. Mladenic, M. Nissim, W. Peters, V. Presutti, and B. Villazón-Terrazas. Selection and integration of reusable components from formal or informal specifications. Technical report, NeOn project deliverable D2.2.1, 2007.
- [SAd⁺07b] 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.
- [SdCd⁺09] Marta Sabou, Guadalupe Aguado de Cea, Mathieu d'Aquin, Enrico Daga, Holger Lewen, Elena Montiel, Valentina Presutti, and Mari del Carmen Suárez-Figueroa. D2.2.3 methods and tools for the evaluation and selection of knowledge components. NeOn Project Deliverable D2.2.3, The Open University, FEB 2009.
- [SFBG⁺07] M. C. Suárez-Figueroa, S. Brockmans, A. Gangemi, A. Gómez-Pérez, J. Lehmann, H. Lewen, V. Presutti, and M. Sabou. Neon modelling components. Technical report, NeOn project deliverable D5.1.1, 2007.
- [SFdCB⁺07] M. C. Suárez-Figueroa, G. Aguado de Cea, C. Buil, C. Caracciolo, M. Dzbor, A. Gómez-Pérez, G. Herrero, H. Lewen, E. Montiel-Ponsoda, and V. Presutti.S. NeOn Development Process and Ontology Life Cycle. Technical report, NeOn project deliverable D5.3.1, 2007.
- [SFdCB⁺09] M. C. Suárez-Figueroa, G. Aguado de Cea, C. Buil, C. Caracciolo, M. Dzbor, A. Gómez-Pérez, G. Herrero, H. Lewen, E. Montiel-Ponsoda, V. Presutti., and B. Villazón-Terrazas. Revision and Extension of the NeOn Methodology for Building Contextualized Ontology Networks. Technical report, NeOn project deliverable D5.4.2, 2009.
- [SGA⁺07] Marta Sabou, Jorge Gracia, Sofia Angeletou, Mathieu d'Aquin, and Enrico Motta. Evaluating the semantic web: A task-based approach. In *ISWC/ASWC*, 2007.

- [Soe95] D. Soergel. Data models for an integrated thesaurus database. *Comatibility and Integration of Order Systems*, 24(3):47–57, 1995.
- [VTAGS⁺08] Boris Villazón-Terrazas, Sofia Angeletou, Andrés García-Silva, Asunción Gómez-Pérez, Diana Maynard, Mari Carmen Suárez-Figueroa, , and Wim Peters. D2.2.2 methods and tools supporting re-engineering. NeOn Project Deliverable D2.2.2, Universidad Politécnica de Madrid, DECEMBER 2008.
- [vZMPR08] R. van Zwol, V. Murdock, L. Garcia Pueyo, and G. Ramirez. Diversifying image search with user generated content. In *Proc. of the ACM Int. Conf. on MIR*, 2008.
- [WD05] Baoning Wu and Brian D. Davison. Identifying link farm spam pages. In Allan Ellis and Tatsuya Hagino, editors, *WWW (Special interest tracks and posters)*, pages 820–829. ACM, 2005.