Deliverable number: D4.1

Deliverable title: # Semantic repository and ontology mapping

Delivery date:	2010/07/31
Submission date:	2010/08/31
Leading beneficiary:	Universidad Politécnica de Madrid (UPM)
Status:	Version 2.0 (final)
Dissemination level:	PU (public)
Authors:	Jorge Gracia, Michal Trna, Esther Lozano, Thanh Tu Nguyen, Asunción Gómez-Pérez , César Montaña, and Jochem Liem

Project number:	231526
Project acronym:	DynaLearn
Project title:	DynaLearn - Engaging and informed tools for learning conceptual system knowledge
Starting date:	February 1st, 2009
Duration:	36 Months
Call identifier:	FP7-ICT-2007-3
Funding scheme:	Collaborative project (STREP)

Abstract

This document discusses the core Semantic Technologies in DynaLearn: i) The *semantic repository*, which supports the online storage and access of qualitative reasoning models, ii) the *grounding* process, which establishes semantic equivalences between the concepts in the models and the concepts in a background knowledge source, and iii) the use of *ontology matching* techniques for discovering similarities between the grounded models, in order to support knowledge-based feedback during the modelling process.

The work described in this document constitutes the core of WP4. This follows the previous work done in WP2 (technical design and architecture) and it is in close relation with WP3 (workbench for conceptual modelling), which handles the interaction with the final user.

Internal Review

- Wouter Beek and Bert Bredeweg, Human Computer Studies Laboratory, University of Amsterdam (UvA), The Netherlands.
- Andreas Zitek, Institute of Hydrobiology and Aquatic Ecosystem Management, University of Natural Resources and Applied Life Sciences (BOKU), Austria.

Acknowledgements

We would like to thank Paulo, Isabella, and Pedro (from University of Brasilia) for having created the golden standard for our model alignment experiment. We thank Andreas, Michaela, Petya, Richard, Ian, Dror, Paulo, Isabella, and Gustavo for their kind help with the grounding evaluation (special thanks to Andreas and Michaela for their detailed comments on the quality of the DBpedia groundings). Thanks also to our internal reviewers for their corrections and useful comments.

Document History

Version	Modification(s)	Date	Author(s)
0.1	Initial draft describing grounding	2010/03/25	Thanh Tu Nguyen
0.2	Repository	2010/06/02	Jorge Gracia
0.3	Ontology matching (initial scheme)	2010/06/12	Jorge Gracia
0.4	Prototype, authentication	2010/07/13	Michal Trna
0.5	Ontology matching	2010/07/14	Jorge Gracia
0.6	Web services, initial draft of advanced grounding	2010/07/16	Michal Trna
0.7	Introduction, update of grounding	2010/07/17	Jorge Gracia, Esther Lozano, César Montaña
0.8	Abstract, general update	2010/07/18	Jorge Gracia
0.9	Conclusions, discussion, update of grounding	2010/07/19	Jorge Gracia
0.10	Review of grounding	2010/07/19	Esther Lozano
0.11	Update of advanced grounding	2010/07/19	Michal Trna
1.0	Draft for review by partners	2010/07/20	Jorge Gracia
1.1	General review and update	2010/07/21	Michal Trna, Jorge Gracia
1.2	Ranking algorithms	2010/07/21	Michal Trna
1.3	General review and update	2010/07/22	Michal Trna
1.4	General review and update	2010/07/26	Asunción Gómez-Pérez, Jorge Gracia
1.5	Reviewed by partners	2010/08/18	Wouter Beek, Bert Bredeweg, Andreas Zitek
2.0	Final version	2010/08/23	Jorge Gracia

Contents

1. Introduction

The global objective of the Semantic Technology (ST) component in DynaLearn is to support learners in their task of finding and comparing models and establishing contact with peer-learners to enhance their mutual learning experience. The models created by peers and experts can be stored in the **repository** as either a final product, or as a work in progress (e.g. as part of a course).

We call semantic **grounding** the process of linking terms in models to concepts in a common vocabulary. Grounding transforms the set of models into a networked resource of multilingual scientific data semantically enabled that can be exploited both in scientific and educational contexts. By allowing comparison between models, algorithms can be written to make modelling suggestions based on other models. Furthermore, when reusing model parts, knowledge can be more gracefully integrated, as equivalent knowledge already existing in a model can be reused. For finding these pieces of common information, **ontology matching** techniques can be applied to explore the similarities among models, with the purpose of getting valuable feedback during the model construction process.

Figure 1 shows and overview of the role of the semantic techniques in the model construction process in DynaLearn.

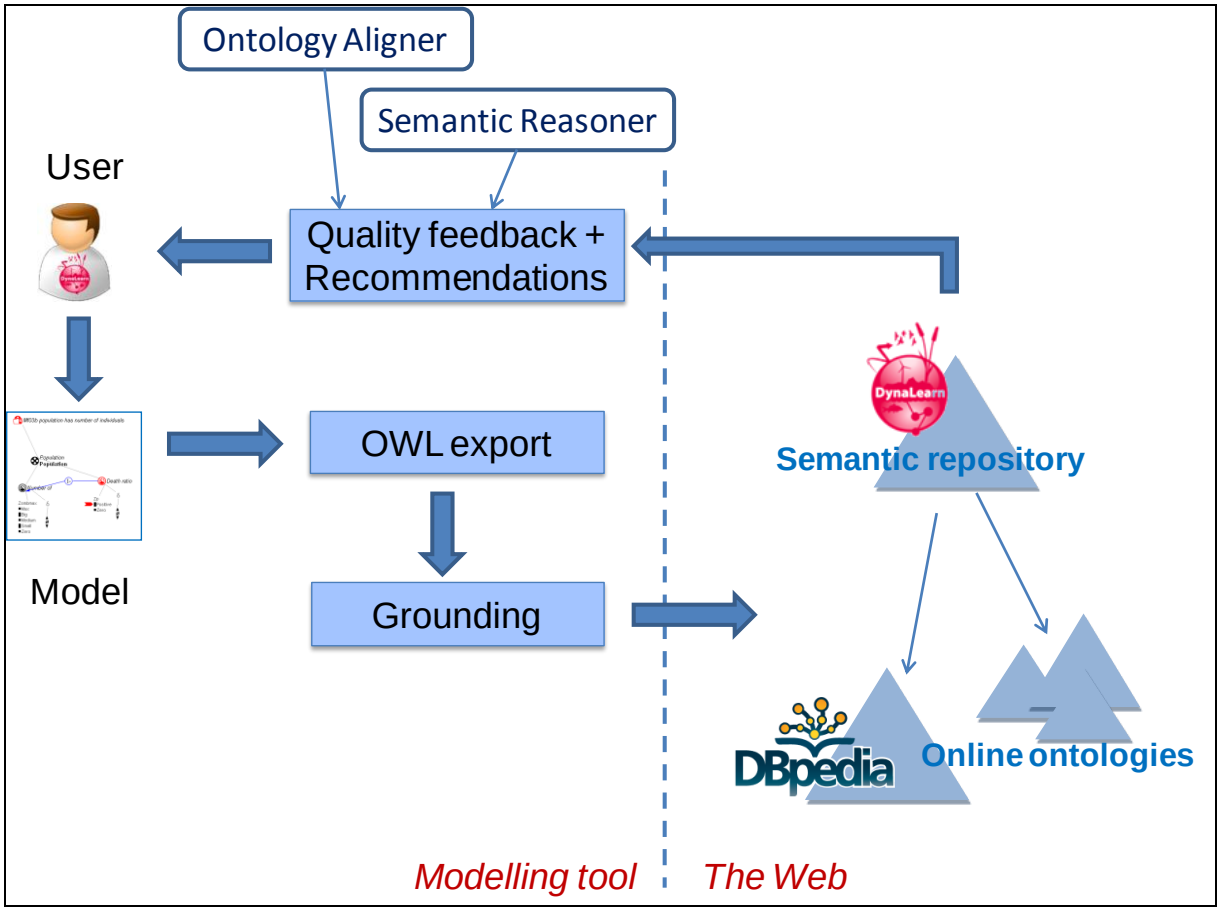

Figure 1: Overview of the approach.

The following steps are identified in the figure:

1. *OWL Export*. The conceptual knowledge contained in QR models is extracted and expressed in OWL, to facilitate their ontology-based description. This process is described in detail in Deliverable D3.2.

2. *Grounding*. The terms from the QR models are linked into external vocabularies (i.e., DBpedia and other online ontologies). These grounded models are stored in the semantic repository.

3. *Quality feedback and recommendations*. Alignment and reasoning techniques are applied to discover similarities and dissimilarities among models and, based on that, to enrich the modelling process with adequate feedback and suggestions for knowledge reuse. Quality feedback is treated in Deliverable D4.2. Also collaborative filtering algorithms will be explored to provide recommendations (to be described in Deliverables D4.3 and D4.4).

The output of this process is a networked pull of online aligned conceptual models (expressed as ontologies) anchored to common vocabularies, and representing specific scientific domains. This has the potential of being a valuable Web resource for scientific progress in general and for semantic guided learning in particular.

In this document we describe the infrastructure required to store the grounded models into an external repository, as well as the basic semantic techniques that support the grounding process in DynaLearn (regardless the user language). We also explore the applicability of ontology matching techniques to discover similarities among QR models, for the final purpose of getting valuable feedback.

The rest of this document is organized as follows. First, the semantic repository is presented in Section 2. In Section 3, the process of semantic grounding is discussed. Section 4 presents the use of ontology mapping techniques in DynaLearn. Section 5 contains the conclusions, and Section 6 finishes the document with some additional discussion. Finally, Appendix A contains some technical details describing how to access the repository and how the grounding process is granted by means of Web services.

2. Semantic repository

The goal of the semantic repository is to store and provide access to the ontological (and non ontological) resources required by the ST component of DynaLearn. Some of these resources are well supported by a traditional relational database, while others need a richer environment that includes semantic capabilities. In order to make this feasible, the following research question is raised:

Among all available semantic storage systems, which one is the most suitable for DynaLearn?

In this section we start by briefly reviewing the different resources that the semantic repository contains. Then, we describe the study we carried out in order to answer the above research question. Finally, we give some details of the user management system. Some implementation details of the repository can be found in Appendix A.

2.1. Components of the semantic repository

The resources supported by the semantic repository include:

- *QR models*, stored in the repository as OWL ontologies by the Conceptual Modelling (CM) component (responsible for the translation of QR models into OWL).
- *Common vocabulary*, or the set of already grounded model terms. This represents the common conceptualization contained in the pool of grounded models. Notice that some of the grounded terms point at external resources (e.g., DBpedia [2]) while other terms, not found in external resources, are stored in the so called *ontology of anchor terms* (see Section 3).
- *Thesauri*. The common vocabulary is complemented by two thesauri: First, a synonym thesaurus, including the different terms used to refer to the same (grounded) term. Second, a number of language thesauri (provided by DBpedia), one per each DynaLearn target language, with translations of each grounded term (when available).
- *User management system*. The needed infrastructure to support user identification and profiles (teachers, learners, etc.), as well as the restrictions on the functionalities that they can access, are also handled by the semantic repository.

The thesauri and the user management system are well supported by a relational database[1], while the other components of the repository (QR models and common vocabulary) need a richer semantic framework that includes certain semantic capabilities not available in traditional relational databases (such as reasoning or semantic querying). The latter will be discussed in the next section.

More details about the common vocabulary (including the ontology of anchor terms), and the thesauri are discussed in Section 3, in which the process of semantic grounding is presented, while the user management systems is presented in Section 2.4

[11] We use mySQL (http://www.mysql.com/) in our current prototype.

2.2. Requirements for a semantic storage system in DynaLearn

Many features can be studied in order to choose a particular tool that supports our semantic repository. We focus, however, on some dimensions that are crucial for the particular application scenario that we devise in DynaLearn:

1. **OWL capabilities**. The QR models are exported to OWL by the CM module, therefore it is required that the storage system is able to deal with this language.

2. **SPARQL capabilities**. SPARQL [21] is an extended technique for querying semantic information on the Web. Concerning our system, we devise the design and use of SPARQL-based patterns and templates, in order to retrieve information from stored QR models for recommendation and ontology-based feedback purposes. Therefore, the ability of using SPARQL is a mandatory requirement for the storage system used in DynaLearn.

3. **Reasoning**. Semantic reasoning techniques are a requirement in the project, in order to discover new non-declared knowledge in the models by applying certain entailment rules. It is expected that the applied inference level will not be high. A trade-off between performance and inference level will become necessary, taking into account that the potential volume of semantic content will be high; and that the higher the expressivity level, the longer the reasoning time.

4. **Scalability**. It is expected that the system grows in number of users and models with time. To begin with, it would be able to allocate at least the expected 35 expert models (about 7 models per partner). By the end of the project more than 100 expert models will be handled, besides models from learners (a precise estimation of the average model size and the approximate number of models added by learners will be possible in future stages of the project).

5. **Performance**. There will be some semantic functionalities in DynaLearn which admit an off-line processing (e.g., classifying all existent models in certain dimensions). Nevertheless, many others will require a run-time response (e.g., making suggestions during model construction). And other might even happen concurrently (e.g., various students in the same class uploading their models at the same time). Therefore, a good behaviour in response time is also desirable for our purposes. As with scalability, this requirement will become more important as the project progresses and the number of users increases.

6. **Licensing**. In order to be consistent with the open source license policy of DynaLearn [11], we will give priority to the use of free and open source tools, always that they are supported by a significant community and come with well documented APIs and source code. Nevertheless, such systems could be replaced in the future by other solutions if the exploitation plan of DynaLearn's bundled software requires it.

These basic guidelines are intended to help us in the choice of a suitable semantic platform at this stage of prototyping.

2.3. Comparative study of semantic storage systems

Without being exhaustive, we summarize in this section some of the most extensively used RDF storage and retrieval systems. In order to compare them, we have focused on the features that are relevant for the purposes of DynaLearn, as it was discussed in Section 2.2.

2.3.1. Main available systems

In Table 1, we show the compared tools and their main features.

Tool	Author	Storage modes	Inference?	OWL compliant?	SPARQL capable?	Licensing	Comments
AllegroGraph	Franz Inc.	File-based	YES (natively: RDFS++ and Prolog rules; admits external reasoner)	YES	YES	Free version limited by a 50 million triples maximum	
BigOWLIM	Ontotext	File-based	YES (RDFS, OWL DLP, OWL Horst)	YES	YES	Commercial license	Built on top of Sesame
Boca	IBM	Database	NO	NO	NO	Open source	No updates since February 2007
Jena	HP	Memory File-based Database	YES (transitive, RDFS, subsets of OWL-DL; admits external reasoner)	YES	YES	Open source	discontinued by HP since October 2009
Mulgara	Paul Gearon	Database	YES (RDFS and user-defined rules)	NO	YES	Open source	
Oracle 11g	Oracle	Database	YES (RDFS++, OWLSIF, OWLPRIME, and user-defined rules; admits external reasoner)	YES	YES	Commercial license	
OWL API	Univ. of Man-chester	Memory	YES (external reasoner)	YES	NO	Open source	
OWLDB	J. Henss	Database	YES (external reasoner)	YES	NO	Open source	Based on OWL API
OWLgres	Clark & Parsia	Database	YES (DL-lite OWL2)	YES	YES	Commercial license for proprietary or commercial uses	
Sesame	Aduna	Memory File-based Database	YES (RDFS)	NO	YES	Open source	
SOR	IBM	Database	YES (OWL DLP)	YES	YES	Commercial license	

SwiftOWLIM	Ontotext	Memory	YES (RDFS, OWL DLP, OWL Horst)	YES	YES	Open source	Built on top of Sesame
SWI-Prolog	UVA (among others)	Memory	YES (Prolog rules)	YES	NO	Open source	

Table 1: Comparative of RDF repositories.

Table 2 contains the web pages of all the tools mentioned in this comparative, where additional information and downloads can be found.

Tool	Web page
AllegroGraph	http://www.franz.com/agraph/allegrograph/
Boca	http://ibm-slrp.sourceforge.net/wiki/index.php/BocaUsersGuide-2.x
Jena	http://jena.sourceforge.net/index.html
Mulgara	http://mulgara.org/index.html
Oracle 11g	http://www.oracle.com/technology/tech/semantic_technologies/index.html
OWL API	http://owlapi.sourceforge.net/index.html
OWLDB	http://owldb.sourceforge.net/
OWLgres	http://pellet.owldl.com/owlgres
Sesame	http://www.openrdf.org/
SOR	http://www.alphaworks.ibm.com/tech/semanticstk
SwiftOWLIM BigOWLIM	http://www.ontotext.com/owlim/
SWI-Prolog	http://www.swi-prolog.org/

Table 2: Web pages of the compared RDF repositories.

There are some other systems in addition to the ones presented above (e.g., Hawk[2]) however not included here owing to its limited support and documentation at the time of writing this.

As we can see from Table 1, only two systems accomplish the following:

- are open source
- admit inference
- are OWL compliant
- admit SPARQL queries

[2] http://swat.cse.lehigh.edu/downloads/hawk.html

Such systems are SwiftOWLIM and Jena. In the following we provide more details about these systems, as well as about Sesame (on which OWLIM is based).

Jena

Jena[3] is a Java framework for building Semantic Web applications. It provides a programmatic environment for RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine. The Jena Framework includes:

- an RDF API
- reading and writing RDF in RDF/XML, N3 and N-Triples
- an OWL API
- in-memory and persistent storage
- SPARQL query engine

Sesame

Sesame[4] is an open source framework for storage, inference and querying of RDF data. It can be deployed on top of a variety of storage systems (relational databases, in-memory, file systems, keyword indexers, etc.), and offers a large scale of tools to developers to leverage the power of RDF and RDF Schema, such as a flexible access API, which supports both local and remote (through HTTP or RMI) access, and several query languages, such as SeRQL.

OWLIM

OWLIM[5] is a high-performance semantic repository developed in Java. It is packaged as a Storage and Inference Layer (SAIL) for the Sesame RDF framework. OWLIM is based on TRREE – a native RDF database and rule-entailment engine.

The semantics, supported by OWLIM, can be configured through rule-set definition and selection. The most expressive pre-defined rule-set combines unconstrained RDFS with most of OWL Lite (as indicated on the OWL fragments map).

OWLIM is available in several versions, tailored to meet different requirements as discussed in the version map. The two major flavours of OWLIM are:

- SwiftOWLIM, performing reasoning and query evaluation in-memory, while a reliable persistence strategy assures data preservation and consistency.
- BigOWLIM, operating with file-based indices, which allows it to scale to billions of statements even on desktop machines.

[3] http://jena.sourceforge.net/index.html

[4] http://www.openrdf.org/

[5] http://www.ontotext.com/owlim/

2.3.2. Comparative studies in the literature

There are some works in the literature which aims to provide comparisons between existent RDF repositories and establish benchmarks and test cases that enable such comparisons [12,13,3,15,19].

Although none of these works is conclusively answering the question "which RDF storage system is the best?" they provide some interesting implications that we can take into account in our study.

In [12,13] the authors compare four systems, two memory-based and two based on persistent storage, by loading and querying OWL ontologies. According the analysis they provide, database-based systems are better for large data sets. On the contrary, memory-based ones are better when the size is relatively small.

The work presented in [3] builds a benchmark for the comparison of OWL reasoners. This gives us an idea of the scope of the different reasoning systems that we can use in a RDF repository, either natively (e.g., Sesame) or externally connected (e.g., Jena). According to [3]:

1) Reasoners that employ a simple rule engine (e.g., Sesame, OWLIM) scale very well for large A-Boxes, but are in principle very limited to lightweight language fragments.

2) Classical tableau-based reasoners (e.g., Pellet [23], Racer [14]) scale well for complex T-Box reasoning tasks, but are limited with respect to their support for large A-Boxes.

3) The reasoning techniques based on reduction to disjunctive datalog, as implemented in KAON2[6], scale well for large A-Boxes, while at the same time they support a rich language fragment.

In [15], the state of the art of RDF storage and retrieval systems is reviewed. The authors have observed that most RDF stores are not really specialized database systems for RDF data but rather an intelligent middleware that wraps existing database technology. Besides providing special support for the graph data model that is characteristic for RDF data, the main functionality provided by this middleware is support for ontological reasoning. An observation that can be made in connection with these two main functions is the fact that almost all systems rely on relational databases that provide very limited support with respect to the data modelling and reasoning. There are very little approaches that aim to delegate some of these aspects to the storage model as well by using deductive or object oriented database technologies.

Regarding future developments in RDF storage technologies, two trends are identified in [15]. The first one is the extension of existing systems to more expressive representation languages. In this context, rule languages are the most promising candidates because it has been shown that rule-based reasoning has the potential to scale to very large data sets whereas ontological reasoning based on description logics shows serious limitations when large numbers of instances are involved. The other major direction of development concerns the scalability of RDF infrastructures to the Web scale.

Finally, here are some conclusions from the work on ontology benchmark carried out in [19]:

- Memory-based RDF repositories can answer queries more quickly than database-based ones, but its scalability is relatively poor.

- The use of different underlying databases affects the performance of the system.

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

- Using mature DL reasoners (e.g., Pellet) gives good results for T-Box inference. The combination of DL reasoners for T-Box inference and rules for A-Box inference is a promising approach.

2.3.3. Discussion

As we already mentioned in Section 2.3.1, only two RDF repositories out of the 12 studied ones are open source, OWL compliant, and SPARQL compliant; namely Jena and SwiftOWLIM. Although SwiftOWLIM showed a better performance than Jena in our in-house experiments, the studies in the literature have shown a really poor scalability (see Section 2.3.2).

Regarding the application scenario in DynaLearn, we consider that scalability is at least as critical as time response. In fact, we expect that the system will grow with time, reaching a point at which the use of memory-based storage only (as in SwiftOWLIM) is no longer feasible. Therefore, a system with a good performance at a small scale but not able to store a large amount of models is not useful for our purposes. From this point of view, Jena is a better solution than SwiftOWLIM.

Some works [13] have reported their concerns about the use of Jena because of its limited performance. Nevertheless, they commonly used the Jena built-in reasoner, while Jena admits the use of other external reasoners, which typically lead to better performance results in T-Box inference. Furthermore, from our experience, we have found out that the best results with Jena require a certain affinity with the tool and a fine tuning of its parameters.

In spite of our concerns about Jena's performance, there is no clear alternative tool that fulfils these requirements: scalable, open source, OWL and SPARQL compliant, and with reasoning capabilities. Furthermore, Jena comes with a complete and well documented API, and it is well supported by a large community of users. For these reasons, we have chosen **Jena** as the tool for developing our Semantic Repository *during the prototyping phase of DynaLearn* (owing to the fact that it is open source, is OWL and SPAQL capable, admits permanent storage, and benefits from semantic reasoning) . The decision about the final tool that will be used for the exploitation phase of the project (that might include commercial ones) is postponed until the project reaches a more mature stage.

Any future replacement of Jena by another platform will benefit from the fact that many of them have Jena-like interfaces or implement adapters to Jena which facilitate the migration. This is the case of Oracle 11g or AllegroGraph, for example.

As future actions, we plan to evaluate the performance and scalability of the platform in a real-usage scenario, as soon as there is a critical mass of models in the repository to enable this study (around M22, where many evaluations in the context of WP7 are due). This is expected to further support our initial selection of Jena or initiate a new round of evaluation of available alternatives.

2.4. User management

Users are required to authenticate in order to access the Semantic Repository (as well as any other functionality in the Semantic Technology component). A User Management System (UMS) as part of the Semantic Technology was developed in order to facilitate the task of creation and management of the user accounts. This application takes the form of a web application, it is written in PHP and stores the user's profile information in a relational database. User accounts are divided into four roles: administrator, teacher, domain expert, and learner. Each of the roles has a precisely specified set of rights. Without going too much into details, their roles can be described as follows:

Administrator

An administrator has the right to manipulate all the attributes of other accounts. He is explicitly forbidden to access the Semantic Technology from the workbench. Thus, no models can be created under an administrator's account, and its sphere of access is entirely inside of the UMS.

Teacher

A teacher can access the UMS and can create new accounts for learners. A teacher is allowed to use features provided by the Semantic Technology component from the workbench. A teacher can choose to assign a TA (Teachable Agent) flag to a model and thus allow learners to use the specified model when operating with the teachable agent in the workbench.

Domain expert

A domain expert has similar rights as a teacher. Domain experts, however, cannot create new accounts for learners.

Learner

A learner is not allowed to access the UMS. A learner can upload new models and can load all the models that were previously stored under his/her account. Under this role it is also possible to load any model with a set TA flag as a source of background knowledge. Learners are not allowed to watch the content of these models though.

2.5. Summary

The semantic repository stores and provides access to the ontological (and non ontological) resources required by DynaLearn, comprising: *QR models* (expressed in OWL language), *Common vocabulary* (or the set of already grounded model terms, which includes the *ontology of anchor terms*), *Thesauri* (both synonym and linguistic thesauri), and *User management system*.

In this section we have described the study we carried out to select the most suitable semantic storage system that (jointly with a traditional relational database) supports our semantic repository. The conclusion of this study was that Jena semantic framework is the option that best fit the DynaLearn requirements. Some basic notions of the user management system have also been provided in this section.

3. Semantic grounding

In the context of DynaLearn, users (both learners and teachers) need to ground terms in well-formed background vocabularies while constructing models. This ensures lexical and semantic correctness of those terms, as well as facilitates the interoperability among models at terminological level. Two types of grounding are foreseen, which correspond to two different levels of granularity: term grounding and model grounding.

Figure 2 illustrates two model fragments put in relation by grounding their terms into an external resource (DBpedia in the example). Although they were created separately, the model terms ("population", "number of", etc.) point to common background ontology terms (*http://dbpedia.org/resource/Population, http://dbpedia.org/resource/Size*, etc.) even if expert and student utilized different vocabulary ("death", "death rate").

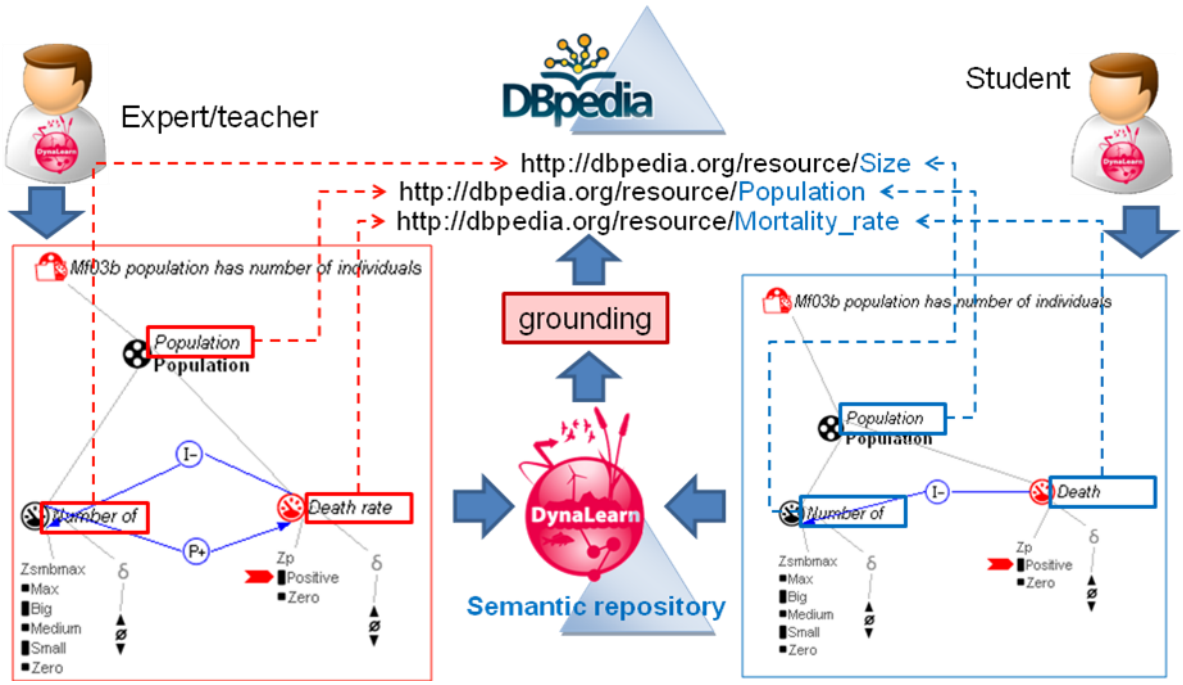

Figure 2: A teacher and student models grounded a common vocabulary.

In this section we review the potential benefits of semantic grounding, as well as the scope and types of grounding. The experiments we carried out to study the coverage and suitability of the different sources of background knowledge are also presented. Finally, we detail the grounding process and algorithms.

3.1. Benefits of Grounding

By grounding a model, we are able to bridge the gap between the loosely and imprecise terminology used by a modeller and the well-defined semantics of an ontology. This facilitates interoperability among models or model fragments. Benefits following from this include:

1. In an educational context, a teacher might restrict the vocabulary used by the learner to the knowledge contained in a certain domain ontology, thus speeding up the training period required to learn that vocabulary.

2. New knowledge can be inferred using standard semantic reasoning techniques. For example, let us suppose that entities *whale* and *mammal* in a QR model are grounded to equivalent terms of the same background ontology. If this ontology asserts that *whale* is a subclass of *mammal*, then the same relationship can be inferred for the entities in the model. Other relations not explicitly declared in the model can be also inferred (such as *whale* is an *animal*).

3. Inconsistencies and contradictions between models can be detected. Besides semantic inconsistencies (which can be discovered by applying a reasoner), other modelling issues can be detected. For example, suppose that a model asserts that the increasing *size* of a *population* increases the *demand of natural resources* of that *population*, while another model establishes the opposite effect, that is, a growing *size* would decrease the *demand of natural resources*. If we are able to establish that both models are referring to the same concepts (*size, population, natural resources*, etc.), the contradiction between the shared concepts can be discovered and pointed out.

4. Additional knowledge and resources can be incorporated into the system. For example, DBpedia contains rich multilingual textual descriptions, links to pictures and web pages, etc. as part of a term description. This information can be imported if the term is grounded on that knowledge source, and shown to the user in the modelling tool.

Most of the previous features are exploited in DynaLearn for enabling knowledge-based feedback, as it is described in Deliverable D4.2.

3.2. Scope of Grounding

Among all model ingredient types in the models assembled in DynaLearn*, which ones would be beneficial to ground?* The QR ingredient types present in DynaLearn can be classified in two groups [18]: “aggregate” ingredients (model fragments and scenarios) and “building blocks” (the rest). We will focus on the later group for grounding (which includes entities, quantities, configurations, agents, etc.) since they can be more easily identified with well-defined ontological classes or properties.

“Aggregate” ingredients do not represent single concepts but how single concepts interact amongst themselves and thus aim to describe a more complex process. This is also reflected in the way “aggregate” ingredients are named: typically with long and complex labels. For example, some scenarios and model fragments from QR models are named “single population full causal growth dynamics”, “what affects quality of sustainability plans”, “scientific involvement config”, “Assume river and delta temperature correspond”. There are, however, other ones described by simpler nouns, such as the “River” model fragment in a model about “algal bloom”[7]. Nevertheless, these single-word names for model fragments are a minority and, typically, their semantics does not correspond to the general concept associated with that name. E.g., the model fragment “River”, in the mentioned example, does not describe what a river is, but the relationships between nutrients, water temperature, etc. in an aquatic ecosystem. Indeed, the same model defines “River” also as an entity which, unlike its homonym model fragment, is intended to represent the general concept of river.

[7] http://hcs.science.uva.nl/QRM/models/CioacaEtAl2009.zip

Out of the group of “building blocks”, we will focus mainly on those types with a richer structure and more easily identifiable with concepts and properties in ontologies, over those that are mostly informative labels (attributes, assumptions) and those representing data types (quantity spaces).

Therefore, the default set of ingredients that more naturally admit grounding is composed of: *entities, quantities, configurations,* and *agents*. This situation does not discard the possibility of grounding other ingredient types as well. Nevertheless, this has to be done with caution, and the final decision postponed until the evaluation experiments planned for WP7 will be done, which is expected to support the final decision (on which terms are more suitable for grounding) with empirical evidence.

3.3. Types of Grounding

From an architectural perspective [1], a user request for grounding is always conducted by the CM component in DynaLearn, which is also responsible of showing the grounding results and handling the user interaction. We identify two operating modes:

- **Single term grounding** (see Figure 3). A request is made from the CM to the ST component to ground a single model term (identified by its URI). A set of ontology terms (from the background ontologies) is returned, characterized by their URIs and some additional information (label, description, synonyms, etc.). Finally, the user has to confirm the grounding, which is eventually saved in the system.

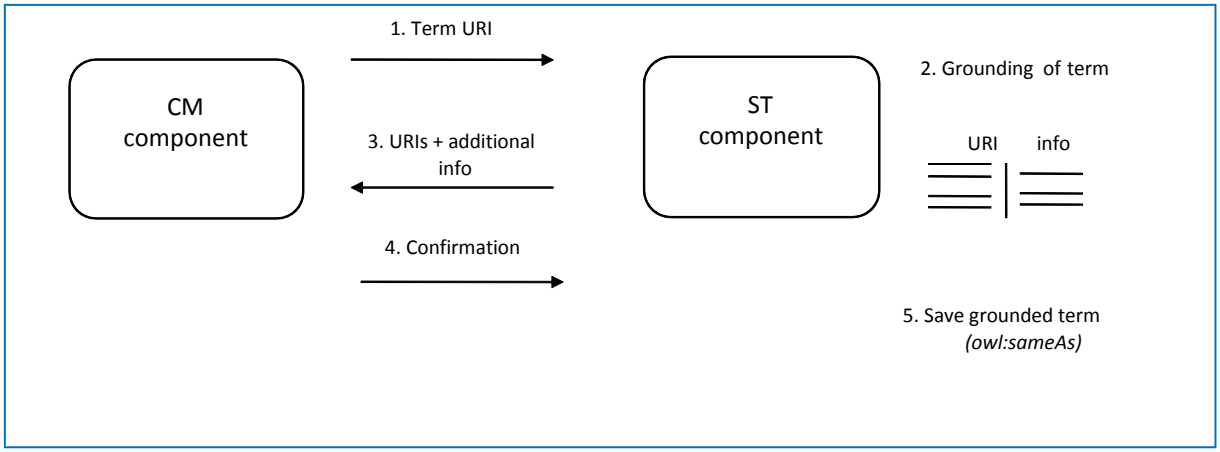

Figure 3: Single term grounding.

- **Model grounding** (see Figure 4). In this case, all terms belonging to a model are grounded with a single request. The whole model is sent to the ST component, which extracts the model terms that are ungrounded, and applies the single grounding process to all of them.

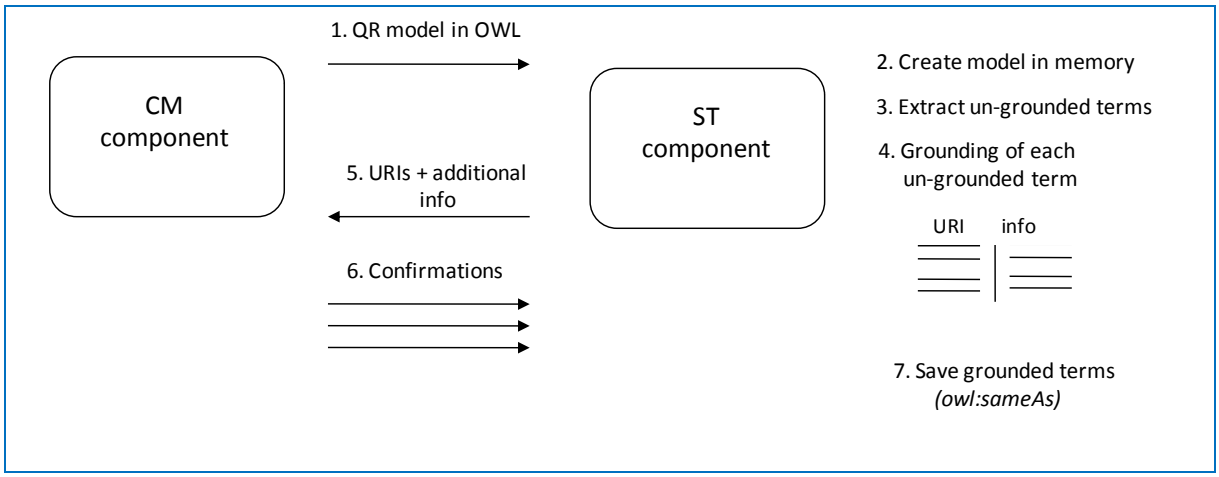

Figure 4: Model grounding.

3.4. Evaluation of external knowledge sources

In order to find out what is the most suitable ontology to be used as a source of background knowledge for grounding, we have performed an experiment to study the coverage level of different ontologies and resources. We understand coverage as the *proportion of terms from certain domain vocabularies that such resources describe semantically*. We have studied also the presence of multilingual labels in the resources.

In a real-world scenario of usage, the QR models are constructed on the basis of specific domain vocabularies. Therefore, we have focused in our experiment on a set of domain glossaries in environmental science developed by partner universities[8] for the DynaLearn project. Each glossary consists of a set of words which covers seven topics: *Earth systems and resources, the living world, human population, land and water use, energy resources and consumption, pollution,* and *global changes*. We merged those glossaries and removed duplicated labels, obtaining a dataset of 1686 labels in English. This unified dataset was used to explore the coverage of knowledge sources of different types: lexical resources such as WordNet [20], common knowledge ontologies such as DBpedia [2] and OpenCyc[9], and the large amount of online ontologies accessible in Watson [7]. Description of the resources follows:

- **DBpedia** is a community effort to extract information from Wikipedia[10] and to make this information available on the Web. The DBpedia knowledge base describes more than 3.4 million concepts, in the time of writing this, out of which 1.5 million are classified in a consistent ontology, and consists of over 1 billion pieces of information (RDF triples).

- **OpenCyc**, the open source version of the Cyc technology, a reasoning engine with a large, multi-contextual knowledge base. It contains hundreds of thousands of English terms along with millions of assertions relating the terms to each other.

[8] University of Brasília (Brazil), Tel Aviv University (Israel), University of Hull (United Kingdom), Bulgarian Academy of Sciences (Bulgaria), and University of Natural Resources, and Applied Life Sciences (Austria). See http://hcs.science.uva.nl/projects/DynaLearn/internal/Wiki/index.php?n=DynaLearn.ContentModels

[9] http://www.opencyc.org

[10] http://en.wikipedia.org/wiki/

- **WordNet**, an electronic lexical database in English. In its version 3.0, WordNet contains over 155,000 words organized in over 117,000 synsets (synonym sets) for a total of around 207,000 word-sense pairs. It provides short, general definitions, and offers various semantic relations between these synonym sets.

- **Watson**, a gateway to the Semantic Web which makes use of a set of dedicated crawlers to collect online semantic content. Watson provides various functionalities for searching and exploring online semantic documents, ontologies, and ontological terms, including keyword search, metadata retrieval, the exploration of ontological term descriptions and formal query facilities.

3.4.1. Coverage Experiment

The first step of the experiment consisted in searching each word of the input dataset on each of the above external resources. Minor adaptations of the data to the input requirements of each resource were needed (e.g., we noticed that DBpedia and OpenCyc need capitalized search terms to operate, or that Watson does not accept the space character as a separator of compound words). The search mode was "exact match". A set of ontology terms from each resource was retrieved (if available), intended to semantically describe the input word. Table 3 shows the different coverage degrees obtained on each resource[11]. The coverage ratio means the proportion of input words out of the initial set of 1686 English words for which a semantic description was found on the resource. Also the multilingual capabilities of each source are shown in the table.

External Resource	Multilingual	Coverage Ratio
DBpedia	Yes	**72%**
OpenCyc	No	69%
WordNet	No	48%
Watson	Yes	47%

Table 3 Coverage results in different knowledge sources.

The immediate conclusion from the given results is that **DBpedia** has a better coverage than the other resources, closely followed by OpenCyc, for the utilized domain specific vocabularies.

We have analysed the uncovered cases in the experiment, noticing that most of them corresponded to complex multi-word terms (e.g., "cultural habit", "distributed water governance"). There was also a reduced amount of spelling errors in the glossaries (e.g., "fiter feeding" for "filter feeding") and some terms that do exist in the resource but in another syntactic variation (e.g., "meandering" for "meander"). In such cases (misspelling errors and variations), the grounding is assisted by services like Yahoo Spelling suggestion[12]. In order to measure that effect, we repeated the experiment with DBpedia but augmented by searching for alternative suggested forms when the term was not found in its initial form. The new coverage of using DBpedia + Yahoo Spelling Suggestion service rose to a **78%**.

[11] The experimental data can be found in http://delicias.dia.fi.upm.es/~jgracia/experiments/dynalearn/groundingcoverage.html

[12] http://developer.yahoo.com/search/web/V1/spellingSuggestion.html

Although all the input data of the experiment was in English, we also explored the presence of multilingual labels in DBpedia for the terms involved in the experiment. Among all the English terms that were found on DBpedia, we were able to find translations of many of them into 15 different languages, in a ratio that ranges from the 48% of terms in Danish to the 72% in German. It shows that DBpedia supports multilingualism to a certain extent, in a degree which is strongly dependent on the target language (see Table 4).

Language	Translations	Language	Translations
Danish	48%	Norwegian	49%
Spanish	64%	Polish	59%
German	72%	Portuguese	58%
French	70%	Russian	56%
Italian	59%	Swedish	56%
Japanese	63%	Chinese	49%
Dutch	61%	Finnish	50%

Table 4 Ratio of translations into other languages in DBpedia for the studied set of English labels.

WordNet and OpenCyc support only English. Watson, on the contrary, indexes any semantic information it finds on the Web, no matter the language. Nevertheless, the use of English to describe ontologies and semantic content is still dominant on the Web (82% of all ontologies that use a particular language are in English[13]). Moreover, although ontologies in more than 20 languages can be found in Watson, only 2% of them are multilingual.

3.4.2. Human-based assessment of the DBpedia results

Though the reached 78% coverage indicates that DBpedia covers the studied domain specific terminologies well, another question arises: *Are the proposed DBpedia-based groundings acceptable according to human opinion or not?* In order to answer that, we randomly selected 909 terms covered by DBpedia from the same glossaries used in the previous experiments.

We asked 8 evaluators (experts in different fields of environmental science) to assess the correctness of the possible meanings given by DBpedia. Each evaluator assessed between 200 and 300 terms. The grounding of each term was double-evaluated.

We counted as positive groundings those terms for which there was at least one suitable meaning among the list of DBpedia results for such a term. We define **accuracy** as *the amount of positive groundings divided by the number of evaluated groundings*. The obtained average accuracy in the experiment was **83%**. The observed inter-evaluator agreement was 85%, and Cohen's kappa [6] of inter-evaluator agreement was 0.47, which can be considered as "moderate" and gives us an idea of the difficulty of the task (if another expert does not fully agree with me, why a computer should?).

[13] http://watson.kmi.open.ac.uk/blog/2007/11/20/1195580640000.html

3.4.3. Combined coverage with several sources

Finally, although DBpedia exhibits a good coverage, we wondered whether it can be further improved with the addition of other resources. Results of combining DBpedia with the other resources can be found in Table 5.

Multiple External Resources	Coverage Ratio
DBpedia and OpenCyc	87%
DBpedia and Watson	73%
DBpedia and WordNet	72%
DBpedia, OpenCyc, Watson and WordNet	88%

Table 5: Coverage obtained by combining multiple external resources.

From the data we conclude that the combined use of OpenCyc and DBpedia increases the coverage significantly (while the further addition of other sources has only a minor effect).

3.4.4. Discussion

The results of the grounding experiment show a high coverage degree of DBpedia (78%) when used to ground domain specific terminologies, as well as a high accuracy (85%) of the covered terms according to human opinion. These results support the use of DBpedia as preferred source of knowledge for the grounding of the vocabulary involved in QR modelling.

Notice that although the coverage of OpenCyc and DBpedia are comparable (see Table 3), there are another reason for choosing DBpedia, such as its multilingual capabilities: DBpedia contains data in up to 92 languages while OpenCyc is monolingual in English, thus reducing its potential usage in a multilingual modelling environment. Nevertheless, the combined use of both ontologies further improves the coverage of English terms up to 87%, as we have found out empirically. Thus, we will complement the use of DBpedia with OpenCyc though for English labels only.

3.5. Grounding process

In this section we detail the grounding process, i.e. the task of linking a term from any QR model to a semantically equivalent and well-accepted term in an external ontology.

3.5.1. State diagram

Figure 5 presents the three different states that we devise in the process of grounding. The state “ungrounded” is the initial state in which the term is created, or in which the term is reverted to after a grounding deletion or a proposal rejection. When the system obtains a candidate grounding for a term, this is in state “to be revised”. When the user confirms the grounding, it changes to the “grounded” state.

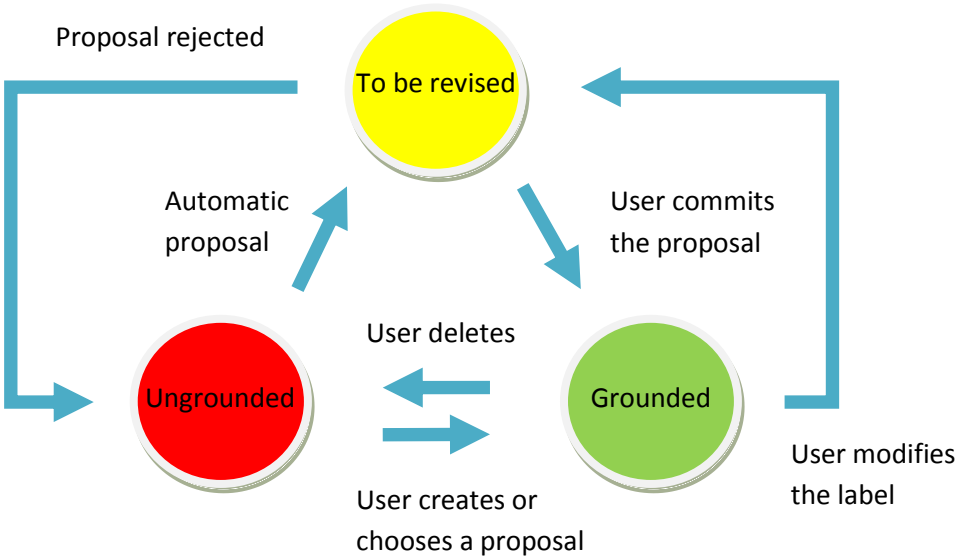

Figure 5: State diagram presenting the dynamics of the grounding.

3.5.2. Utilized Resources

We mention here the main (external) resources accessed during the grounding process:

- DBpedia. We access DBpedia both as a source of possible meanings for the grounded terms as well as a source of multilingual information (which enables the grounding in several languages). Also the synonyms (equivalent labels) given by DBpedia during the grounding process are captured to feed our synonym thesaurus (see Section 2).
- WordNet. We exploit its morphological processor for normalization purposes of English labels. Furthermore, WordNet is an additional source of synonyms in the system, which are stored in the synonym thesaurus.
- Yahoo Spelling Suggestion. This is used to suggest corrected variants of labels ("Did you mean?") when they are not found in the background knowledge resource.

3.5.3. Algorithm for grounding a model term

The algorithm for grounding examines syntactic coincidences between the label of the model term to be grounded and the terms in the background ontology, expanded by the use of synonyms and label suggestion services. For instance, when the user adds an entity "car" to the model, the system proposes, among other senses, the term *http://dbpedia.org/resource/Car* as a candidate term for grounding.

In the semantic repository is maintained a thesaurus of synonyms, based on synonym labels that are extracted from the sources of background knowledge, enriched with information available in the repository. E.g., "automobile" might be equivalent to "car" in the synonym thesaurus, so *http://dbpedia.org/resource/Car* could also be proposed for the grounding of an entity labelled "automobile".

Formalization of the algorithm for grounding of a model term follows:

Input:

- model term to be grounded, represented by its label
- language of the label
- type of ingredient of the model term: "entity", "quantity"...

Output:

- list of grounding proposals represented by their URIs and other descriptive information

Process:

1. **Reuse of previous groundings**. Previous groundings are searched in the semantic repository. If found, they are added to the list of grounding proposals.
2. **Access to external sources of background knowledge**. The external resources (i.e., DBpedia) are accessed for discovering possible matches between the label of the model term and the labels of the background ontology terms. If found, the list of discovered ontological terms are added to the list of possible groundings.
3. If the list of grounding proposals is non empty:
    3.1. **Ranking of grounding proposals**. The list of grounding proposals is ranked according to the relevance of each candidate grounding and taking into account the context of the term in the model.
    3.2. The list is sent to the workbench and shown to the user, who chooses the most suitable grounding (if any). If the user confirms a grounding, the process finishes.
4. If not grounding proposals are discovered in 1) or 2), or those discovered are unsatisfactory to the user, then:
    4.1. **Process unsatisfactory groundings.** The workbench triggers a method offering possible actions:
        a) Suggestion of corrections to the label are provided.
        b) Suggestion of the addition of new model terms to the vocabulary are provided (e.g., when a multi-word is not groundable but one of its component is).
        c) A new anchor term is created.
        d) The term remains ungrounded.
    4.2. If, as result of 4.1 new grounding proposals are found, then step 3 is repeated, otherwise the process finishes.

The main steps of the above algorithm (those written in boldface) are described with more detail in the following paragraphs.

Reuse of previous groundings

The first step in the grounding process is to check whether we can reuse previous groundings or not. This can be done by exploring the *cache* of grounded terms allocated in the semantic repository.

The cache of grounded terms consists of a table that contains all the grounded terms from the models stored in the repository. Information about the grounded term is stored into the cache, including the author of the grounding and a link to the resource used for grounding (either DBpedia resource or anchor term). The process first compares the term with already grounded terms from that cache; this is done by comparing the labels of the terms. Finally, all the results obtained from this process are proposed as candidate groundings for the current model term.

Access to external background knowledge sources

After searching for previous groundings, the system searches for groundings on external resources (i.e., in DBpedia). Before searching DBpedia through the Web, the system searches first in the cache of external groundings allocated in the semantic repository. All the results obtained from DBpedia are always stored in the cache (both effectively grounded and not). This information will be available with a lower latency the next time a model term with the same label is grounded. If no suitable result is found in the cache of groundings, then the term is searched directly on DBpedia.

Ranking of grounding proposals

In order to help the user in the task of choosing a proper grounding, a ranking of the grounding proposals is carried out. Communication of this ranking to the CM component allows CM component to provide the user with the most relevant items on the highest positions of the list of proposals in the user interface. The following methods (which combine state of the art techniques) have been devised in order to achieve such a behaviour:

Context comparison

Context of the model term is extracted from the model and compared to the context of the candidate grounding. In case of model term, the context is formed by a set of labels of the model terms related to the considered term in the model. For the candidate grounding, the context is produced from the textual description, which is available both in DBpedia and in the internal repository of groundings. Both sources are processed to a set of words with frequencies and a technique of **vector space modelling** [22], which is in our case *term frequency-inverse document frequency* method (tf-idf), is applied in order to produce a single number capturing the relative similarity of these contexts. It is assumed that labels with similar context bear similar meanings and therefore candidate groundings with high similarity to the original label should be promoted.

String based distance

Applying an algorithm to compute how different two strings are provides a mean of comparison between the label of the model term and the label of the proposed grounding. In our case we use the Levenshtein distance [17]. The candidate groundings with high string-based distance are suppressed.

Techniques presented above provide independent metrics of which a final metric is assembled, assigning a weight to each of them in a sum. Concrete numerical values are a matter of experimentation which is planned to take place in the latter stages of the project.

Process unsatisfactory groundings

In this section we discuss techniques used in order to deal with model terms for which no grounding proposals are generated by the first stage of an algorithm of grounding.

Corrections suggested by Yahoo

Yahoo Spelling Suggestion[14] web service provides a suggested spelling correction for a given term. This service is used to assemble a list of suggestions to correct the label in case of unsatisfactory grounding. For example, suppose that we try to ground a label “fiter feeding”, which contains spelling errors (and therefore is not found in DBpedia). The Yahoo Spelling Suggestion service retrieves “filter feeding”, which can be effectively found in DBpedia.

Dealing with labels containing multiple words

A specific process is started in this case, in which important parts of the original label are identified and extracted, in order to use them for enriching vocabulary or as a source of proposal for alternative naming for the model term (see Section 3.8).

Anchor terms

When the user decides to introduce a term as an **anchor term**, a new term is introduced in the repository as a part of a common vocabulary that is built in the DynaLearn. The new term is introduced to a specific namespace (which is http://www.dynalearn.eu/ontologies/AnchorTerm.owl), it has a description given by the user, and is available to be used for future groundings.

3.5.4. Modification of a grounded model term

Model terms can be modified by the user after they are grounded. Change can occur in the context of the model term, e.g. when user binds it to a configuration, or in the proper attributes, e.g. when attributes assigned to it are changed. Most of these changes do not provide a strong justification to question the validity of the previous grounding (if it was valid). When user changes the label of the model term, however, the system invalidates automatically current grounding to state “to be revised” (see Figure 5). In this state, user decides either to confirm the current grounding or to erase it.

3.5.5. Grounding information communicated to the user

A grounding proposal contains following fields:

- URI of the term
- Label of the term
- Description in natural language extracted from the external resource (when available)
- Source: an identifier indicating the provenance of the grounding, i.e. "dbpedia" DBpedia, "anchor" for anchor ontology
- List of Synonyms (if any)

[14] http://developer.yahoo.com/search/web/V1/spellingSuggestion.html

- A list of QR models using this resource for grounding

This is the information communicated to the CM component, though not everything is shown to the final user (who typically does not need to deal with URIs). Figure 6 and Figure 7 illustrate the type of information shown to the user when grounding a model and when introducing an anchor term, respectively[15].

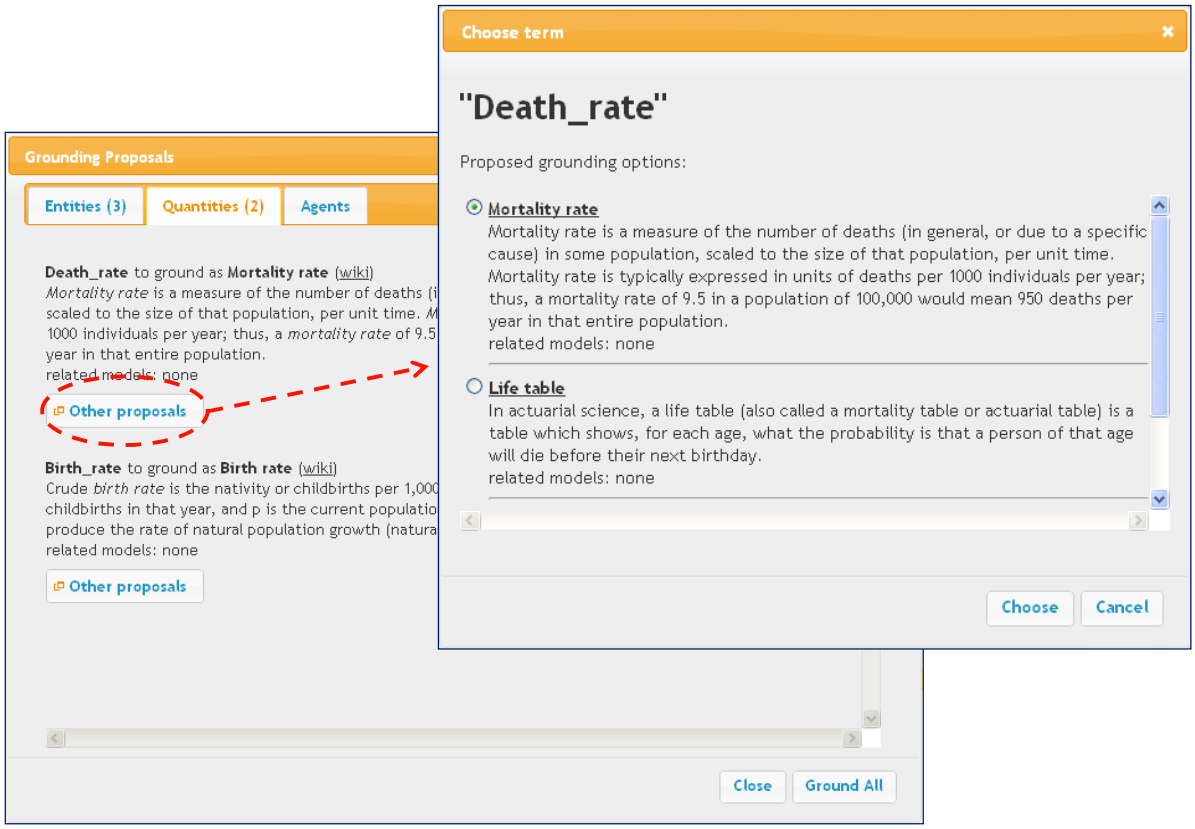

Figure 6: Example of model grounding.

[15] Notice that these pictures correspond to the initial prototype and might differ from the final implementation.

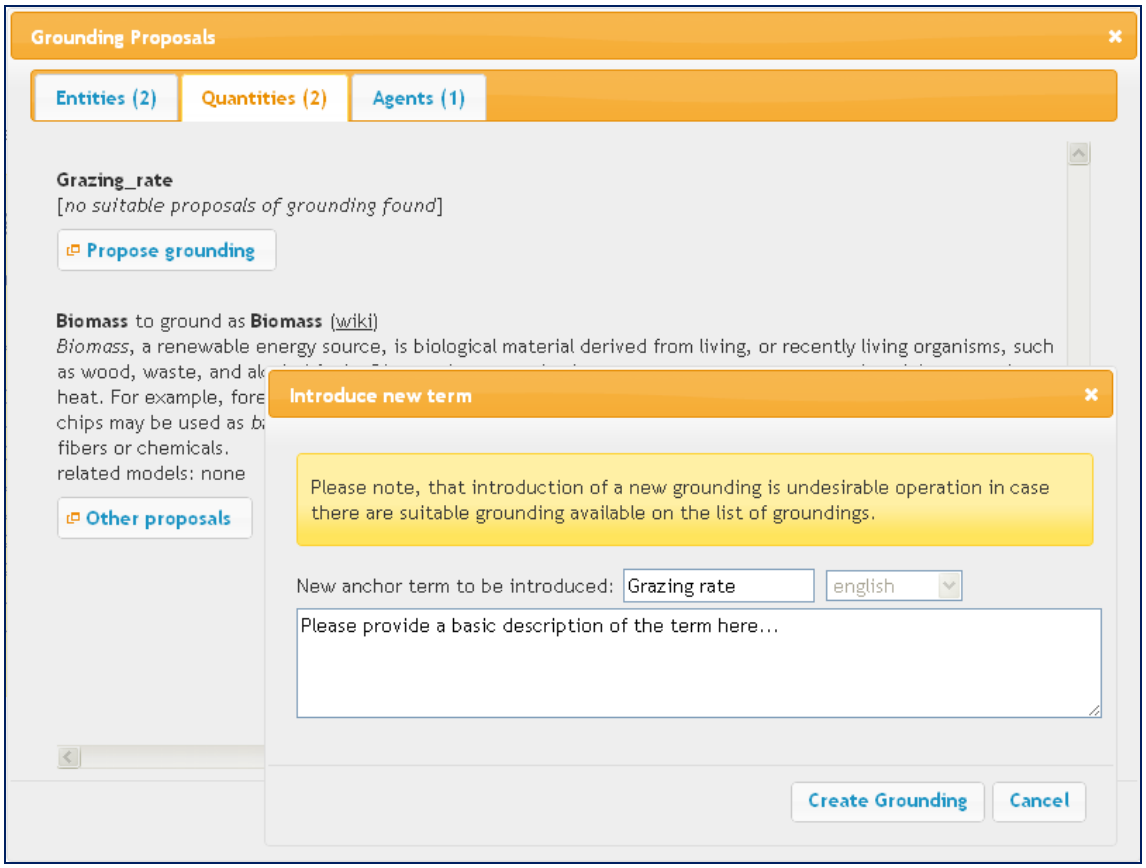

Figure 7: Example of anchor term creation.

3.6. Storage of groundings

As a result of the grounding process, we have to store all the correspondences created during the process for future use. Groundings are stored in two ways at the same time:

- In the cache of groundings (stored in the repository as entries in a relational database).
- In the QR model itself. In that case, groundings to external resources and to the repository of groundings are represented by the presence of the *sameAs* property defined in OWL, see the following example:

```
<owl:Class rdf:about = "&qrm;Filter_feeding">
    …
    <owl:sameAs df:resource="http://dbpedia.org/resource/Filter_feeder"/>
</owl:Class>
```

By using this OWL construct, we have introduced a clear and efficient mean how to distinguish resources that have grounding from those that do not have one.

3.7. Multilingualism

Although a full cross-lingual interoperation among models is out of the scope of DynaLearn, the use of different languages in DynaLearn for modelling is still feasible with the semantic techniques that have been presented so far in this document.

As it was mentioned in Section 2, we include in the semantic repository a language thesaurus containing the translation of grounded terms into the target languages of DynaLearn. More specifically, we have developed that thesaurus on the basis of the multilingual information that DBpedia serves. Indeed, although DBpedia is extracted from the English Wikipedia only, there are many pages on the English Wikipedia that contains explicit multilingual information or pointers to wikipedias in other languages. This information is captured also in DBpedia.

We have extracted this multilingual information from DBpedia, put it in a relational database (the linguistic thesaurus), and incorporated it into the grounding process. As result, the grounding process is sensitive to the specific language which has to be specified as an input parameter (see Section 3.5).

The considered languages in DynaLearn are: English, Portuguese, German, Bulgarian, and Spanish. Notice, however, that the maximum coverage in DBpedia is attainable in English only. For other languages, the coverage degree varies (see Section 3.4).

Finally, although cross-lingual interoperation is not a fundamental requirement in DynaLearn, a basic form of interoperation is still possible by exploiting the multilingual labels that the linguistic thesaurus provides, thus enabling that two terms in different languages can be grounded on the same ontology term. E.g., two entities “Tree” and “Arbol”[16], used in different models (the first written in English and the second in Spanish) can be grounded on the same URL *http://dbpedia.org/resource/Tree*, as long as “Arbol” belongs to the set of multilingual labels of this term.

3.8. Grounding of model terms with multi-word labels

In this section, we discuss issues which we face when labels used for naming model terms consist of multiple words (we call them *multi-words*). We also propose different ways to partially alleviate the problems they cause, enriching the process of searching for grounding proposals.

3.8.1. Description of the problem

As it was mentioned above, the basic grounding functionality in DynaLearn looks for syntactic coincidences in a source of background knowledge (DBpedia) of the labels that describe model terms. If the term is not found, different variations are searched instead (e.g., by applying a spelling suggestion system).

The coverage of DBpedia with this simple technique is good, resulting in a 78% when applied to a set of 1686 terms of the environmental domain, as we have seen in Section 3.4. Nevertheless, if we inspect closely the 22% of remaining uncovered terms in this experiment (those without candidate groundings in DBpedia), we find that the large majority of them have multi-word labels (19% of the

[16] “Árbol” is a Spanish word for “tree”.

total terms), while they constitute less than one half of the terms in the experiment (44%). Figure 8 illustrates this.

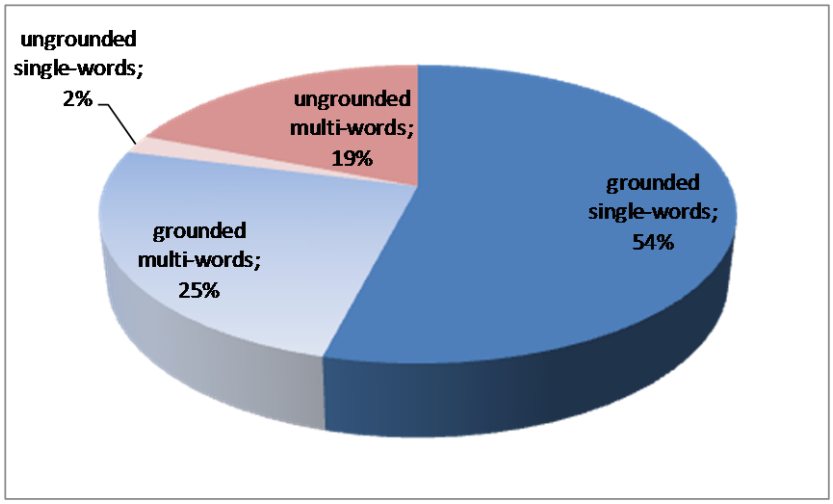

Figure 8: Ratio of un/grounded multi-words (dataset 1: terms from glossaries of the environmental domain).

We carried out a similar experiment but with 367 terms[17] extracted from "real" (made by experts) Garp3 [5] models[18], finding out that this effect (a majority of terms with multi-word labels are not grounded) is even more remarkable in a real-usage scenario, as it is shown in Figure 9.

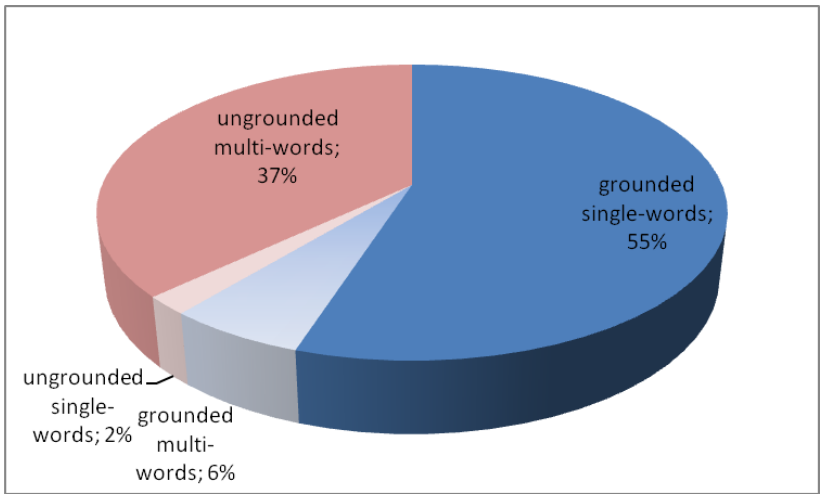

Figure 9: Ratio of un/grounded multi-words (dataset 2: terms from Garp3 models).

Table 6 shows some examples of multi-word labels (just for illustrative purposes), both with and without candidate groundings in DBpedia.

Term	Hit In Dbpedia
Reed warbler	YES
Water industry	YES
Social development	YES
Symbiont one	NO

[17] Considering entities, quantities, configurations, and agents.

[18] Which can be found in http://hcs.science.uva.nl/QRM/models/

Open population	NO
Flows into	NO

Table 6: Sample of multi-words.

The proportion of multi-word terms in the amount of ungrounded ones is really noticeable (85% and 95% respectively in the above mentioned experiments). Nevertheless, even if we cannot directly ground many multi-words, we propose certain processing techniques in order to suggest alternative groundings as well as the inclusion of related terms in the QR model, as we will see in the following paragraphs.

3.8.2. Multi-word processing

This section describes the general algorithm to process multi-word terms during the grounding process in DynaLearn. The general idea of this algorithm is to explore whether the multi-word term can be grounded. If not, it is split into its component words and a tagger is applied to obtain a part of speech (PoS) of each component word. Notice that the process is independent of the language, as far as we use a language independent tagger[19].

Let us call *cluster* a contiguous nonempty subsequence of a sequence of words. E.g. sequence of words “A B C” contains clusters: “A B C“, “A B”, “B C”, “A”, “B” and “C”; subsequences “A C” and “” are not clusters according to this definition.

Algorithm

FOR EACH label *l* in the QR model DO:

Apply **grounding process** to *l*:

IF there are no candidate groundings for *l* AND *l* is a multi-word term THEN

Initialize an empty set *R* of candidate clusters for grounding

Apply **tagger** in order to obtain the PoS of each component word

Split *l* into clusters: find all clusters $\{C_i\}$ such that each C_i represents a sequence of JJ, JJS, NN, NNS, NP, NPS[20] containing always at least one noun (sequences of adjectives only are not allowed); add all C_i to the set R

Suggest adding *l* as anchor term

FOR all elements *r* from *R* DO:

IF *r* groundable:

Suggest grounding *r* and enriching model by adding *r* to the model

Suggest using *r* instead of *l* as the label of the model term

[19] We use TreeTagger (http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/) in our prototype.

[20] JJ represents an adjective, JJS represents an adjective in plural, NN represents a common noun, NNS represents a common noun in plural, NP represents a proper noun, and NPS represents a proper noun in plural.

END IF

END FOR

Assign **specificity score** (see later) to each cluster and sort the suggestions by specificity ranking of the cluster that they belong to

END IF

END FOR

Notice that we suggest in the algorithm enriching the model with new terms. However, we do not imply any concrete place in the model where the new term should be placed.

3.8.3. Specificity ranking algorithm

The specificity ranking produces a score for each cluster produced from the original label. It is a metric which supports promoting more semantically specific clusters on the list of suggested actions, in order to offer a user with the most relevant resources first, while penalizing short and vague terms. A simple algorithm was implemented in order to count the score.

Algorithm

If cluster equals the original label, assign 0 to the score.

In case it is not, iterate over the words in the cluster:

For each adjective in the cluster, add 1 point to the score.

For each common noun in the cluster, add 2 points to the score.

For each proper noun in the cluster, add 3 points to the score.

Sort the set of clusters according to the score, highest values being first.

Example of ranking

An example of input and resulting output follows.

Input: label “change of energy flow” (quantity)
Tagging: change(common noun)of(preposition)energy(common noun)flow(common noun)
Output, suggested (alternative) actions:

“change of energy flow” (specificity score: 0)
- add “change of energy flow” as anchor term

“energy flow” (specificity score: 4)
- if groundable, ground “energy flow” + add it to the model
- if groundable, rename label to “energy flow”

“energy” (specificity score: 2)
- if groundable, ground “energy” + add it to the model
- if groundable, rename label to “energy”

“flow” (specificity score: 2)
- if groundable, ground “flow” + add it to the model

- if groundable, rename label to “flow”

“change” (specificity score: 2)

- if groundable, ground “change” + add it to the model
- if groundable, rename label to “change”

3.8.4. Evaluation

We applied the previous algorithm to the same data utilized for the experiment described in Section 3.8.1. As result, we observed that for the first dataset (terms from environmental science glossaries), alternative groundings or suggestions for the large majority of ungrounded multi-words were found (18% of the total of terms), as it is shown in Figure 10, where “alternative multi-words” represents the amount of multi-words not grounded initially but for which the algorithm founds one or more groundable alternatives.

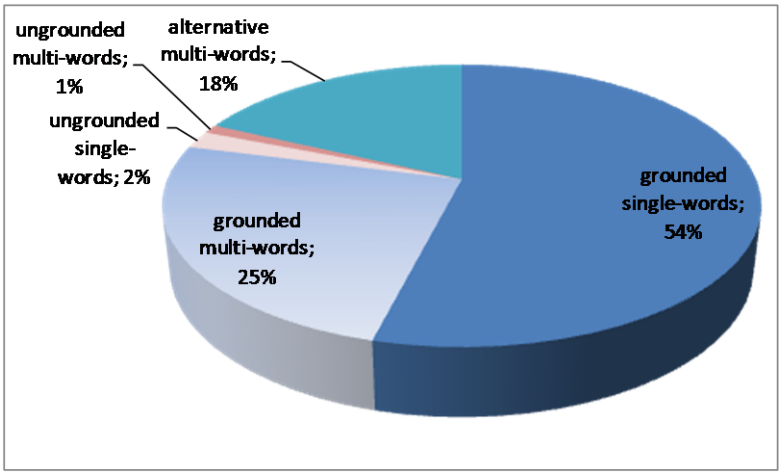

Figure 10: Ratio of un/grounded/alternative multi-words (dataset 1: terms from glossaries).

Figure 11 shows the result of the equivalent analysis for the second dataset (those terms coming from Garp3 models). We see from the figure that a 34% of terms are multi-words and have alternative groundings even if they could not been initially grounded.

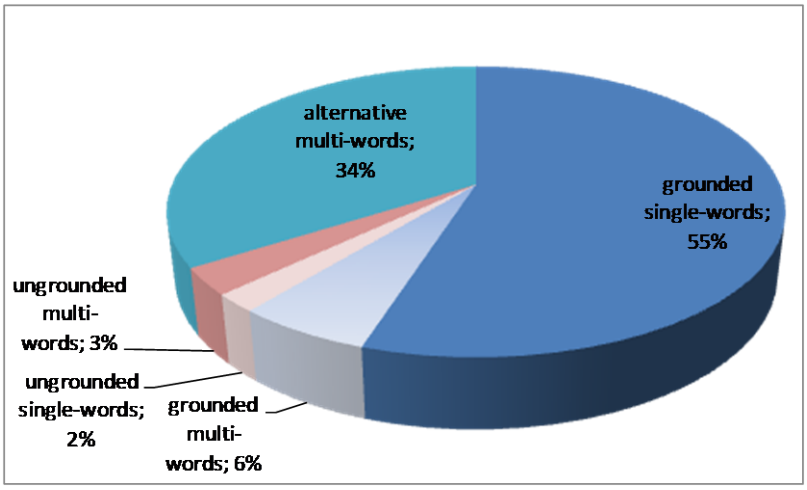

Figure 11: Ratio of un/grounded/alternative multi-words (dataset 2: terms from Garp3 models).

These results support the usefulness of this technique to improve the quality of the grounding process. They can potentially increase the coverage of grounding, as well as enrich the modelling process by suggesting the addition of new terms into the model.

3.9. Summary

This section has detailed the grounding process in DynaLearn, required to ensure lexical and semantic correctness of the utilized QR model terms, as well as to facilitate the interoperability among models at terminological level. We have detailed our grounding experiments with several sources of background knowledge, showing how DBpedia exhibits the best behaviour (well supported by human opinion), therefore constituting our preferred knowledge source.

The algorithmic details of the grounding process have also been presented, as well as some notes on multilingualism and the specific treatments needed for enriching the grounding of multi-words.

4. Ontology Matching

As we have discussed in Section 1, we need techniques to reconcile different QR models in order to analyse the similarities and differences between them and, based on that, provide useful feedback during the modelling construction. To that end, we use well-established ontology matching techniques in DynaLearn.

Ontology Matching (OM) (referred also as *ontology mapping* or *ontology alignment*) is the task of discovering correspondences between terms from different ontologies [9] (or models, in our case). It is symbolised in Figure 12.

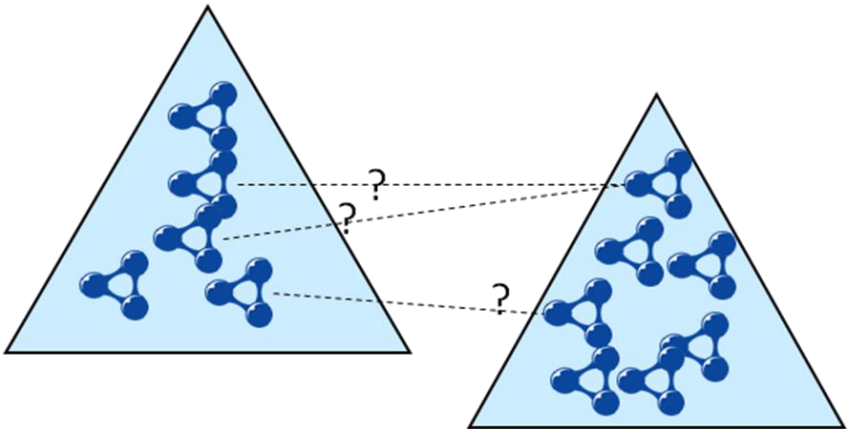

Figure 12: The problem of ontology matching.

The general scheme of an ontology matching system is represented in Figure 13, where O and O' represent the input ontologies, A is an initial set of known correspondences (optional), and A' is the resultant alignment (the found correspondences) [9]:

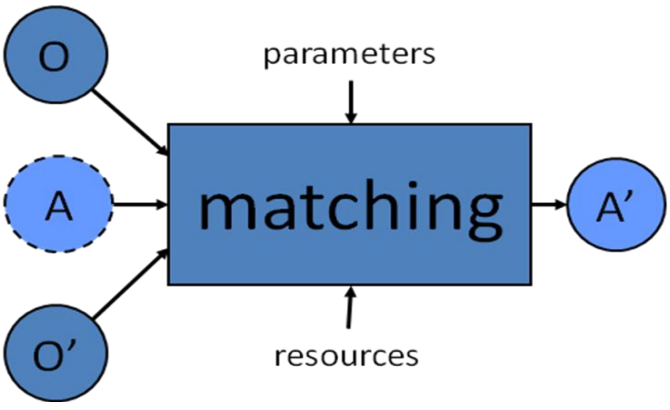

Figure 13: Scheme of an OM system.

4.1. The role of Ontology Matching in DynaLearn

QR models can be expressed as ontologies without loss of meaning (see [18]). Therefore, traditional OM techniques can be adopted to compare models as far as they are described in terms of an ontology language (OWL in our case).

In particular, there are two components in DynaLearn where this comparison between ontological models can play an important role:

1. During *semantic grounding*: To better select a suitable ontology term associated with the grounded model term.
2. During *ontology-based feedback*: To align a user model to other reference models.

Let us briefly review each of these two cases.

4.1.1. OM during semantic grounding

Traditional OM methods [9] (two whole ontologies are received as input) are difficult to apply at *run-time* "as they are" during the grounding process owing to the huge size of the background ontologies (DBpedia, OpenCyc, etc.). In fact, these techniques typically perform pairwise comparisons between all the terms in the ontologies (or their partitions) to discover semantic equivalences, resulting in a very time consuming process for large ontologies.

Nevertheless, the application in this context of certain lightweight *syntax-based* OM is still feasible, by exploring syntactic equivalences, expanded with alternate forms (based on stemming, suggestion services, and synonym expansion). We have adopted this lightweight OM technique approach for the grounding process, as it has been described in Section 3.

4.1.2. OM during Ontology Based Feedback

OM fits more naturally in the process of getting knowledge-based feedback, when comparing models or model fragments to reference models (expert models, teacher models, ...). In fact, comparisons are favoured by the smaller size of models (with respect to background ontologies), and by the similar nature of the compared ontologies (e.g., they share the same domain, the same meta-model, etc.).

A more detailed description of how OM techniques are used for getting ontology-based feedback is out of the scope of this document and can be found in Deliverable 4.2.

4.2. Evaluation of OM techniques for QR model comparisons

As we have seen before, we are interested in using ontology matching techniques to reconcile QR models produced by different authors modelling similar domains. This is a fundamental part in our task of discovering relevant feedback during the model construction process. Thus, our first interest is to answer this question:

Are the state of the art ontology matching techniques suitable for mapping QR models?

Our target is to apply already existent alignment approaches in the context of QR modelling. We have tested the use of two state of the art ontology matching systems: Falcon-AO [16], and CIDER [10]. We chose these two systems owing to their good behaviour in previous Ontology Alignment Evaluation Initiative (OAEI) competitions[21] and to their complementary nature: Falcon is more focused on structure-based matching and incorporates efficient partition-based mechanisms, while CIDER relies on a context-based similarity measure and applies lightweight semantic reasoning.

The evaluation was conducted as follows:

1. First, a golden standard was defined by human experts. They created specific QR models for this experiment and identified semantic equivalences between them.
2. Then, each ontology aligner system was run separately. Each one received the two QR models as input, and produced an alignment file as result, containing the found correspondences between the two models.
3. Finally, each produced alignment was compared to the golden standard, and *precision* and *recall* were computed.

4.2.1. Golden standard

The motivation of defining a golden standard is to make the experiment to a maximum possible extent objective and repeatable. The golden standard was obtained from experts working in the domain [22]. Eight QR models grouped in pairs were chosen, and a reference alignment file was produced for each pair, containing a total of 85 equivalences between the model terms. We expressed the alignments in the Alignment Format [3], to facilitate their later processing.

Particularly, an initial set of four pairs of related models, manually aligned, was provided:

- Case 1 - Social aspects of population growth, v1 vs. v2
- Case 2 - Soil contamination, v1 vs. v2
- Case 3 - Hervibory vs. Predation
- Case 4 - Amensalism vs. Commensalism

Each manual alignment was given in a file containing pairs of equivalent terms. E.g. (from Case 2, expressed in CSV format):

```
1,entity,agro_industry,agro_industry
2,entity,avian_population,soil_community
3,entity,soil,soil
4,quantity,contamination,contamination
5,quantity,Direct_disch_of_ind_waste,leaching_of_wastes
...
```

We translated the original file into the Alignment Format [8] (in RDF) to facilitate its later processing. The ingredient types considered in the experiment were: *entities, quantities*, and c*onfigurations*.

[21] http://oaei.ontologymatching.org/

[22] Three researchers in biological science from University of Brasilia (Brazil). The data can be found in http://delicias.dia.fi.upm.es/~jgracia/experiments/dynalearn/omtechniques.html

4.2.2. Results

After running the experiments, we compared each resultant alignment with the corresponding reference alignment, computing precision and recall as follows:

Precision = num. obtained “gold” mappings / total obtained mappings

Recall = num. obtained “gold” mappings / total “gold” mappings

Our preliminary tests shown issues with the QR model representation when applying Ontology Matching techniques. Particularly, the domain building blocks are not defined by their properties, as is common in ontologies. To alleviate this issue (among others), a supplementary representation of the domain vocabulary accompanies each QR model. The properties of the domain vocabulary are programmatically inferred from the richer model fragment and scenario representations. More details about this reduced representation can be found in [18].

Table 7 shows the result of running the experiment with the above mentioned reduced OWL representation, using with both tools separately (we examined also their combined use, though not discovering significant improvements with respect to the best separated results).

	CIDER			Falcon		
	Precision	**Recall**	**Time [s]**	**Precision**	**Recall**	**Time [s]**
Case 1 (pop.)	1.00	1.00	10.8	0.80	1.00	1.9
Case 2 (soil)	1.00	1.00	9.2	0.79	1.00	2.0
Case 3 (h./p.)	1.00	1.00	4.2	0.63	1.00	1.8
Case 4 (a./c.)	0.67	0.80	4.6	0.44	0.80	1.7
AVERAGE	0.92	0.95	7.2	0.67	0.95	1.9

Table 7: Results of the OM experiment.

Notice that the lower precision given by Falcon is in part owing to the fact that Falcon also aligns the imported ontologies (thus, it aligns the QR vocabulary imported in the models). If the alignment is post-processed and these unnecessary alignments are removed, precision also reaches **92**% (for the reduced representation).

These results illustrate that the use of traditional ontology matching techniques perform well for giving the similarity between QR models. Only minor adaptations are required in order to use such techniques for the purposes of DynaLearn. As future work we plan to run additional tests, and to modify the OM techniques in order to improve its performance.

4.3. Summary

Techniques are needed in DynaLearn to reconcile different QR models in order to analyse the similarities and differences between them and, based on that, provide useful feedback during modelling construction. In this section we have presented the role of ontology matching in DynaLearn, and we have analysed the applicability of two state-of-the-art matching techniques in the context of DynaLearn. Our experimental results support the use of CIDER system to discover alignments between QR models.

5. Conclusion

This deliverable describes the current state and progress on Task 4.1 “Semantic repository and ontology mapping”. First, we have presented the semantic repository in DynaLearn, as well as the study we did to support the selection of a suitable tool for that. Then, we have extensively described the grounding process, which is fundamental for most of the semantic technologies involved in DynaLearn. Our experimentation with the possible sources of background knowledge for grounding is also discussed in the document, and the reasons why we chose DBpedia as preferred source of knowledge for grounding. We have also shown that the application of traditional ontology matching techniques to compare QR models in DynaLearn is feasible. A brief description of the Web services-based interface for the interaction of the Semantic Technologies with other DynaLearn components is available as an appendix of the document.

In summary, the goal of establishing the core semantic technologies in DynaLearn has been completed. The hypothesises we did along the process are well supported by experimentation, as we have shown in this document. Of course, further refinement of our techniques is expected, when more evaluations will be available in future stages of the project.

6. Discussion

There are some open issues to be solved in the remaining of the project that we mention in this section. It is still unclear if all “groundable” model ingredients (entities, quantities, configurations, agents) have to be treated equally during the grounding process, owing to their different nature: Well formed configurations, for instance, describe relations (typically expressed by structure verb + preposition, e.g. “lives in”, “flows into”,...) in which entities take part and therefore consistently display a poor coverage in the external resources, which attention is focused mainly around describing objects and events. One of the solutions we are considering for further experimentation is introducing a predefined vocabulary of configurations in the ontology of anchor terms. Another question that we plan to address is whether grounding of other ingredients would be beneficial for the goals of the project; this problem we plan to resolve by experimentation.

As it has been shown in this document, DBpedia offers a good coverage for the grounding of terminologies in the environmental domain. A human-based assessment evaluation (also reported in this document) confirmed the quality of these groundings. Nevertheless, as result of this experiment, we noticed that the interpretation of some commonly used terms is not satisfactory in DBpedia (e.g., “effect”, “state”, “treatment”, “development”, “acidification”, “shelter”, etc.). As future work we plan to combine the use of DBpedia with OpenCyc (which also exhibits good results in our experiments), in order to maximize the coverage, and also to add more useful definitions to these commonly used terms.

References

1. E. André, N. Bee, R. Bühling, J. M. Gómez-Pérez, M. Häring, J. Liem, F. Linnebank, B. Thanh Tu Nguyen, M. Trna and M. Wißner. B. Bredeweg (ed.). “Technical design and architecture DynaLearn”, EC FP7, STREP Project no. 231526, Deliverable D2.1., 2009.

2. C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. “Dbpedia - a crystallization point for the web of data”. Web Semantics: Science, Services and Agents on the World Wide Web”, 7(3):154–165, September 2009.

3. J. Bock, P. Haase, Q. Ji, and R. Volz. “Benchmarking owl reasoners”. In ARea2008 - Workshop on Advancing Reasoning on the Web: Scalability and Commonsense. June 2008.

4. B. Bredeweg (ed.), “DynaLearn - Engaging and informed tools for learning conceptual system knowledge”, Description of work (DoW), Collaborative project (STREP), Call identifier FP7-ICT-2007-3, Project number 231526, October 9th, 2008.

5. B. Bredeweg, F. Linnebank, A. Bouwer, and J. Liem. “Garp3 - workbench for qualitative modelling and simulation”. Ecological informatics, 4(5-6):263–281, 2009.

6. J. Cohen. “A coefficient of agreement for nominal scales. Educational and Psychological Measurement”, 20(1):37–46, April 1960.

7. M. d’Aquin, C. Baldassarre, L. Gridinoc, S. Angeletou, M. Sabou, and E. Motta. “Characterizing knowledge on the semantic web with Watson”. In 5th International EON Workshop, at ISWC’07, Busan, Korea, November 2007.

8. J. Euzenat, “An API for ontology alignment” in 3rd International Semantic Web Conference (ISWC'04), Hiroshima (Japan). Springer, November 2004.

9. J. Euzenat and P. Shvaiko, “Ontology matching”. Springer-Verlag, 2007.

10. J. Gracia and E. Mena, “Ontology matching with CIDER: Evaluation report for the OAEI 2008”, in Proc. of 3rd Ontology Matching Workshop (OM'08), at ISWC'08, Karlsruhe, Germany, vol. 431. CEUR-WS, October 2008, pp. 140-146.

11. J.M. Gómez-Pérez, P. Salles , M. Wissner, D. Mioduser, B. Bredeweg, R. Noble, Y. Uzunov, A. Zitek, and consortium members. “Dissemination and Communication Plan”. DynaLearn deliverable D8.1, August 2009.

12. Y. Guo, Z. Pan, and J. Heflin. “Choosing the best knowledge base system for large semantic web applications”. In Proceedings of the 13th international World Wide Web Conference on Alternate Track Papers & Posters, New York, NY, USA, May 2004

13. Y. Guo, Z. Pan, and J. Heflin. “An Evaluation of Knowledge Base Systems for Large OWL Datasets”. In Third International Semantic Web Conference, Hiroshima, Japan, LNCS 3298, Spinger, 2004.

14. V. Haarslev and R. Möller, “Description of the RACER System and its Applications”, Proceedings of the International Workshop on Description Logics (DL-2001), Stanford, USA, 1.-3, pp. 132-141, August 2001.

15. A. Hertel, J. Broekstra, and H. Stuckenschmidt. “RDF Storage and Retrieval Systems”. In Handbook of Ontologies. Springer-Verlag, 2008.

16. W. Hu and Y. Qu, “Falcon-AO: A practical ontology matching system”, Journal of Web Semantics, vol. 6, no. 3, pp. 237-239, 2008.

17. V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals”. Cybernetics and Control Theory, 10(8), 707–710, 1966.

18. J. Liem, W. Beek, F. Linnebank, and B. Bredeweg. “API for data and knowledge exchange”, DynaLearn, EC FP7, STREP Project no. 231526, Deliverable D3.2, June 2010.

19. L. Ma, Y. Yang, Z. Qiu, G. Xie, and Yue Pan. “Towards A Complete OWL Ontology Benchmark”. In 3rd European Semantic Web Conference (ESWC2006), June 2006.

20. G. A. Miller. “Wordnet: A lexical database for English”. Communications of the ACM, 38(11):39–41, November 1995.

21. E. Prud'hommeaux and A. Seaborne. “SPARQL Query Language for RDF”. Technical report, W3C Recommendation, January, 2008.

22. V. Raghavan and S. Wong. “A critical analysis of vector space model for information retrieval”. In Journal of the American society for information science 37, 1986.

23. E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, “Pellet: A practical OWL-DL reasoner”, Web Semantics: Science, Services and Agents on the World Wide Web, 5(2):51-53, June 2007.

Appendix A: prototype implementation and Web services

In this section we discuss some implementation details of the Semantic Technology (ST) component, as well as the functionalities provided by the ST component to the Conceptual Modelling (CM) component.

As it was mentioned in the document, we chose Jena semantic web platform as basis for our semantic repository. Nevertheless, other technologies are also involved (web services, relational databases[23], etc.), since we require to cover aspects such as communication with other components in DynaLearn, intermediate caches, user management, etc.

Thus, the implementation of the semantic repository, and of the ST component in general, and its integration in the *DynaLearn workbench* is done as follows. A set of tightly coupled services is deployed in an application container, which is in our case an installation of the last release of Apache Tomcat[24]. In order to facilitate the accessibility to the consumers of the services by a robust and standardized protocol, all the public functionalities of the earlier mentioned services are provided throughout one connection point of Web Services[25]. Implementation of the underlying SOAP[26] communication protocol is handled in the module Apache Axis2[27]. Authorization is required on each request and it is facilitated by Apache Rampart[28] module providing implementation of WS*Security layer[29]. The services provided both on the front-end and back-end of the Semantic Technology component include:

- Access to the semantic repository, which encompasses the functionalities that effectuate the storage and retrieval of the models for both internal purposes and for the DynaLearn workbench. The semantic repository uses Jena Semantic Web Framework, in order to manipulate the models represented in OWL format.
- Accessing and caching mechanisms of the external resources, which are either locally installed or are invoked by the network. Access to external resources is cached in order to provide low average latency of the response time to the modelling workbench on the front-end.
- Access to the ontology of anchor terms.

As a naturally pure provider of services, ST is granted a role of a server, while CM functions as a client or consumer. The technology stack chosen for communication between two components is SOAP/Web Services. This solution is well motivated: Web Services provide standard, transparent and ready-to-use tools and solutions thus allowing for avoiding technical difficulties of dealing with limitations on traffic posed by firewalls and simultaneous access to the server by many clients. In Deliverable 3.2, Section 11, the architecture and communication protocol between ST and CM are described in greater detail.

[23] http://www.mysql.com/

[24] http://tomcat.apache.org/

[25] http://www.webservices.org/

[26] http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[27] http://ws.apache.org/axis2/

[28] http://ws.apache.org/rampart/

[29] http://www.oasis-open.org/committees/security/

The functionalities of the ST component are implemented in Java and published via the aforementioned Web Services API. Apart from the functionalities related to ontology-based feedback, which are in detail described in Deliverable 4.2, following basic functionalities are offered to the consumer (covering areas described in Deliverable 3.2, sections 11.4 and 11.5):

Retrieve model – retrieveModel(String modelID) – Provides consumer with a model from the repository, identified by ID modelID.

Store model – storeModel(String modelID, String xml) – Stores a given model in the repository.

Query model – queryModels(String[] listTerms, String lang) – Produces a list of models related to any of the terms in a given listTerms in language lang.

List models – listModels() – Produces a list of all models available in the repository to the current user.

Grounding proposals for a label – proposeGrounding(String label, String lang, String type) – Provides a list of grounding proposals for a given label of a given type of ingredient type in a given language lang.

Proposals of treatments for a given term – proposeTreatment(final String label, final String lang, final String type) – Provides a list of suggested operation for a given label label. This may contain suggestion to enrich the vocabulary of the model, rename the label or proposal to construct an anchor term of it.

List topics – listTopics(String modelID) – Lists all topics related to a given model in an array of topic labels.

Delete model – deleteModel(String modelID) – Delete model with a given ID modelID from the repository.

Ground model – groundModel(String modelID, String xmlContent) – Generate a list of grounding proposals for a all groundable model terms in a given model.

Create anchor term – createAnchorTerm(String label, String lang) – Generates a record in the repository of anchor terms for a given label.

Delete anchor term – deleteAnchorTerm(String uri) – deletes anchor terms

e-mail: Info@DynaLearn.eu
website: www.DynaLearn.eu