Mark Hall

A Semantic Similarity Measure for Formal Ontologies

(With an application to ontologies of a geographic kind)

Diplomarbeit
zur Erlangung des akademischen Grades
Diplom-Ingenieur

Magisterstudium Informatik (2003)

Alpen-Adria Universität Klagenfurt
Fakultät für Wirtschaftswissenschaften und Informatik

Begutachter:	O. Univ.-Prof. Dr. Johann Eder
Institut:	Institut für Informatik-Systeme

03/2006

Ehrenwörtliche Erklärung

Ich erkläre ehrenwörtlich, dass ich die vorliegende Schrift verfasst und die mit ihr unmittelbar verbundenen Arbeiten selbst durchgeführt habe. Die in der Schrift verwendete Literatur sowie das Ausmaß der mir im gesamten Arbeitsvorgang gewährten Unterstützung sind ausnahmslos angegeben. Die Schrift ist noch keiner anderen Prüfungsbehörde vorgelegt worden.

Word of Honour

I honestly declare that the thesis at hand and all its directly accompanying work have been done by myself. Permission has been obtained for the use of any copyrighted material appearing in this thesis and all such use is clearly acknowledged. The thesis has not been presented to any other examination board.

Mark Hall, Klagenfurt am 23.03.2006

Acknowledgements

I would like to thank all the people who have in any way contributed to this thesis, but I've unfairly singled out a few for honourable mentions.

Peter Mandl for always being there when I needed someone to bounce stupid ideas off and for always offering constructive help and criticism. Prof. Martin Seger for the evaluation work he did for me. Martin Raubal and Michael Lutz for the evaluation work and also for the interesting and educational discussions we had in Münster and Klagenfurt. Angela Schwering for the interesting discussion we had about semantics and cognitive models in Klagenfurt. Mario Lassnig for lots of things such as reading parts of the thesis, giving advice, preparing the gorgeous style that this thesis uses and finally for always knowing all relevant dates and requirements and all the other bureaucratic stuff so that I didn't have to. Also worth mentioning are my trusty Apple PowerBook on which and the ÖBB's trains in which most of this thesis was written.

Finally I would like to thank Nora for the continuing support and for listening to me droning on about this topic for hours on end.

Abstract

The Semantic Web aims to provide a more intelligent web by automatically combining information from different, heterogeneous systems. To overcome the barriers posed by this heterogeneity it is necessary to have some kind of automatic integration system. This system not only needs to provide integration services on the syntactic level, but also has to provide integration of the different semantics. This thesis introduces a model for encoding semantics that is based on cognitive principles. Building on this cognitive model a semantic similarity measure is defined that makes it possible to compare the semantics of two or more data sources in order to provide integration services. To show that the approach is usable in real-world situations it is applied to data from the land-use and land-cover domain. Evaluation of this application proves that the cognitive model and semantic similarity measure provide semantically valid results.

Contents

1 Introduction

1.1 Motivation

The rise of the Internet and other forms of global communication have made it easier to communicate, exchange information and collaborate across national borders. While this has eased and improved all areas of life it has also created additional problems when information systems from different countries, firms or other institutions try to work together. These problems stem from the fact that each of these institutions has in the past created its own information system with its own data structures and interfaces to the rest of the world. If these institutions wish to collaborate then they have to find a way to translate between their data structures and interfaces. This is a time-consuming and difficult process that has to be repeated for every new system that needs to be integrated.

1.1.1 Heterogeneous Data Sources

The difficulties in integrating different systems arise from the fact that the involved groups have slightly different views of the domain, have varying intentions as to what the data structure will be used for and also have different prior experience with building such systems. All these factors combine to create data structures that while describing the same domain are nevertheless different enough to make it hard to combine these data structures into one homogeneous system.

While this has always been true it used to be harder to exchange data since it had to be physically transferred from the data provider to the data user. Integration was then just another small step in the data transfer process. With the rise of the Internet as a communication and collaboration platform accessing remote data became much easier and thus the integration problem more pressing. This is especially true as the future vision of a semantic web[BernEtAl01] requires that systems automatically integrate multiple data sources to intelligently answer user requests.

1.1.2 Data Integration

Data integration is the process of determining which differences exist between the data sources, finding solutions how to overcome them and finally integrating the data. On the data level the differences that can arise have various forms such as

- *Different representations* used for encoding the data. While this seemingly only involves rewriting the data in the other representation it can be problematic when one of the two representation provides constructs that are not available in the other.
- *Different constraints* on the data. Data from one of the two sources may not be valid data if the constraints from the second source are applied to it.
- *Different data types or units of measure* for the same attribute. While this is easily solvable using transformational algorithms this will always lead to a certain loss of information.

The data sources also need to be integrated on the schema level. The problems involved are similar to those described in [Klei01]

- *Differing scope* of concepts means that the concepts have the same name but the sets of individuals that belong to the concepts are not equal.
- *Differing modelling styles* have their roots in the fact that different views of the domain and intentions lead to differing modelling styles.
- *Different terminology* covers such problems as the uses of synonyms and homonyms which make integration harder since equivalent or different names do not immediately imply equality or inequality.

The problems on the schema level are much harder to solve especially automatically since they require an understanding of the semantics of the schemas.

1.2 Objectives

In order to automatically integrate existing data sources it is necessary to also provide a system for the integration of the semantics of the data sources and this in turn requires that the semantics are encoded in a way that can be processed automatically. One way is to use ontologies[Grub93] to describe the semantics of the schemas and then to use these descriptions for the integration process.

This thesis introduces a methodology for encoding semantics in ontologies that aims to maximise the expressiveness of the semantic representation. Building on this semantic representation a semantic similarity algorithm is presented that makes it possible to automatically calculate the semantic similarity between two concepts from two different ontologies. From the results of this semantic similarity comparison the semantically most similar concepts for two ontologies can be determined. This is then used to translate data from one schema into the other schema.

What sets this semantic similarity algorithm apart from existing approaches is that it is grounded in a cognitive model and can at the same time be implemented efficiently. The cognitive grounding guarantees that the results that the algorithm gives are similar to those that would be provided by a human. At the same time the semantic similarity algorithm has been successfully implemented and tested in a system for the integration of land-use and land-cover data. Existing systems either do not have a cognitive grounding or have problems when implemented.

1.3 Terminology

Most of the examples in this thesis and the application described in chapter 5 have been taken from the domain of *land-use and land-cove*r data sets. Land-use and land-cover data sets consist of sets of polygons that cover an area of interest. While *land-use* actually only refers to those areas that are actively used by humans such as agricultural or urban areas and *land-cover* describes the natural areas such as glaciers, forests and rivers in this thesis the term land-use and land-cover will either be written in its abbreviated form as LUC or simply as land-use. The different land-uses that exist in the area of interest are assigned to land-use *categories* which are themselves arranged in a usually hierarchical structure called a land-use *catalogue*.

The semantic similarity algorithm itself works on *ontologies* a term that is defined in chapter 2. These ontologies consist of *concepts* that are also arranged in a hierarchical structure. Thus when the semantics of land-use catalogues are encoded in ontologies the categories become concepts. While this thesis tries to use only the word concepts when dealing with ontologies and categories only for land-use categories when ontologies of land-use catalogues are discussed the terms are sometimes used interchangeably.

1.4 Overview

This thesis is organised as follows. Chapter 2 gives an introduction into ontologies, their foundations in Description Logics and ontology modelling principles. Chapter 3 looks at existing algorithms for automatic data integration and analyses their strengths and weaknesses. In chapter 4 the semantic similarity algorithm is presented together with some information on the cognitive models that form its grounding and also an evaluation of the algorithm's results. An application of the semantic similarity algorithm has been developed and is described in chapter 5. Finally chapter 6 presents conclusions that have been drawn from the development of the semantic similarity algorithm.

2 Ontologies as Semantic Storage

Current computer systems deal with information in a very static manner. The semantics of the data are encoded in the algorithms dealing with the data and in the documentation describing the algorithms and data. They are only accessible to the people who work with and develop the algorithms and data structures. The algorithms themselves have no understanding of the semantics and this limits the flexibility of the algorithms since to change the semantics the algorithms have to be changed. To increase the flexibility and power of these algorithms it is necessary for them to gain an understanding of the semantics of the datas that are involved. This requires that the semantics are specified and stored in a form that is accessible to algorithms and the formalism that this thesis will concentrate one is the ontology as a semantic storage system.

Originally ontology was the study of the existence of entities and is the most fundamental branch of metaphysics. This is important since when the term was acquired by the artificial intelligence and knowledge representation communities the basic meaning was kept but reduced in scope and made less abstract to increase its useability in computer systems. The currently accepted definition is by [Grub93] and [UschGrun96] as a shared formal specification of domain knowledge which contains all important aspects of ontologies as they are understood in the knowledge representation field today

- Most important is that they deal only with *domain knowledge*. This implies that they only describe a limited domain of knowledge. They say nothing about what may or may not be true for entities outside this domain.
- They are *formal* meaning that they are based upon formally specified constructs. This is important since it allows algorithmic use and therefore the use in computer systems.
- They are *specifications* of knowledge. The knowledge described within the ontology does not have to reflect the real world fully, but it is specified as being correct within the scope of the ontology.
- They are *shared*. This is an extension of the specification aspect. The knowledge, which may or may not be true for the real world, is assumed to be true by all those who have agreed upon that knowledge and therefor commit to it. Those applications and users who do not commit to the ontology can ignore the knowledge.

In their dealings with knowledge ontologies restrict themselves to concepts and their relations. Different formalisms and languages have been developed within the ontology field to facilitate this. In this thesis the focus will be on those that have their foundation in the field of Description Logics.

2.1 Description Logics

Description Logics have developed from research started in the 1970s on knowledge representation systems [NardBrac03]. At the time there were two main ideas on how to structure knowledge. The first one was that first-order logic should be used to give a very precise definition to the concepts and relations of entities in the real world. The other approach was influenced by the desire to have a representation that is closer to the way that humans structure knowledge. The second approach mostly took the form of semantic networks [Quil67] or frame based systems [Mins81]. The problem with these systems was that although they were very appealing due to their human cognition centred origins they lacked a precise characterisation of their semantics. Work by [Haye79] led to the realisation that frame systems could be given basic semantics by using first-order logic. Additionally it was found [BrachLev85] that only a restricted subset of first-order logic was necessary and that this restriction had useful implications for the complexity of reasoning on such constructs. After this research in the Description Logics area began under the name of terminological systems with the emphasis on the basic terminological definition. Attention then moved to the concept forming aspects under the name of concept languages and finally with the attention moving to the underlying logical system the name Description Logics (DL) became popular.

2.1.1 Description Logics Languages

Many different sub-languages of Description Logics exist. They vary in their expressiveness and thus also in the complexity of reasoning on them, but all are based on atomic concepts and atomic roles that can be combined into complex descriptions using concept constructors. [SchmSmol91] define the language AL to be the smallest DL language that provides a practical use. In the following definitions A and B shall signify atomic concepts, R atomic roles and C, D concept definitions. The language AL is defined in [BaaderNutt03] as

$$\begin{array}{rcl}
C, D \longrightarrow & A & | \text{ (the atomic concept)} \\
& \top & | \text{ (the universal concept)} \\
& \bot & | \text{ (the bottom concept)} \\
& \neg A & | \text{ (atomic negation)} \\
& C \sqcap D & | \text{ (intersection)} \\
& \forall R.C & | \text{ (value restriction)} \\
& \exists R.\top & | \text{ (limited existential quantification)}
\end{array}$$

As an example of what can be expressed with this language we assume that Water and Flowing are atomic concepts. Based on these two we can now define a River to be Water $\sqcap$ Flowing and a Lake to be Water $\sqcap$ $\neg$Flowing. Additionally with $\forall$ hasWater.$\top$ and $\exists$ hasWater.$\bot$ we can define those water areas that have water and those that are dried out.

The AL language can be extended or restricted by adding or removing constructors. Examples of more restricted languages are FL^- which disallows negation and FL_0 which also removes the limited existential quantification. The following constructs can be added to AL to increase its expressiveness

- Union of concepts $(C \sqcup D)$ is written as U,
- Full existential quantification (∃ R.C) is written as E,
- Number restrictions $\geq n$ and $\leq n$ meaning that the number of concepts filling a property must be at most or at least n written as N,
- Negation of arbitrary concepts $(\neg C)$ written as C.

Thus the AL family of languages consists of $AL[U][E][N][C]$. Not all of these languages are semantically distinct, for example union and full existential quantification can be described by negation and vice versa. This means that all AL languages can be written as $[U][E][N]$ and since $[U][E]$ are equivalent to $[C]$ the family is usually written as $AL[C][N]$.

2.1.2 Formalisms

The basic formalisms for Description Logics are the TBox and the ABox. In the TBox the terminology of the knowledge base is defined, while the ABox contains assertions about individuals in the knowledge base. This thesis will focus on the aspects of the TBox since the remaining parts of this thesis will confine themselves to dealings on terminologies and not individuals.

A TBox contains a set of terminological axioms which define how concepts and roles are related to each other, with the most general terminological axioms being

$$C \subseteq D \text{ and } C \equiv D$$

The first axiom is called inclusion and the second one equality. By restricting the left hand side to an atomic concept the equality axiom can be specialised.

$$\text{River} \equiv \text{Water} \sqcap \text{Flowing}$$

These are called *definitions* and the left side of the equality is called a *symbolic name*. A set of terminological axioms is only a valid TBox if each symbolic name is defined only once. The symbolic names simplify the construction of large knowledge bases since complex definitions can be reused in other definitions. Figure 2.1 shows an example of a TBox. It demonstrates four basic constructors. First a RIVER is defined as the set of those things that are at the same time WATER and FLOWING. Then a LAKE is defined as those things that are WATER but not FLOWING. The next definition adds the handling of properties and the ability to define constraints on their cardinality as in the FOREST definition which says that a FOREST must have at least one tree. Finally the CONIFEROUS

$$\begin{aligned}
\text{River} &\equiv \text{Water } \wedge \text{Flowing}\\
\text{Lake} &\equiv \text{Water } \wedge \neg\text{Flowing}\\
\text{Forest} &\equiv \text{Vegetation} \wedge \geq 1 \text{ hasTree}\\
\text{ConiferousForest} &\equiv \text{Forest } \wedge \exists \text{hasTree.Coniferous}
\end{aligned}$$

Figure 2.1: TBox example

FOREST shows the ability to define restrictions on property ranges when it is defined that at least one tree must be CONIFEROUS.

An ABox would contain information on individuals in the knowledge base. For example

$$\text{River(danube)}$$

defines that the individual danube exists and is an instance of the concept RIVER.

2.1.3 Reasoning

The advantage of using Description Logics in a knowledge base is that its axioms can be translated into first-order logic. This means that inference can be used to make the implicit knowledge contained in the axioms explicit.

Four main inferences exist in Description Logics

- *Satisfiability.* A concept C is satisfiable with respect to a TBox if there is a model of the TBox so that the set of elements of concept C in the model is not empty.
- *Subsumption.* A concept D subsumes a concept C if the set of elements of C is a subset of the set of elements that D describes with respect to a TBox.
- *Equivalence.* Two concepts C and D are equivalent if the sets of elements are equal with respect to a TBox.
- *Disjointness.* Two concepts C and D are disjoint if the intersection of their sets of elements is the empty set.

Of these four inferences satisfiability and subsumption are used most. Satisfiability because when updating a Knowledge Base by adding, removing or changing concepts it is important to know whether the new or changed concept is a valid concept within the TBox and also whether other existing concepts have become invalid through the addition or change. For example if you add the concept

$$\text{BrokenRiver} \equiv \text{River} \sqcap \neg\text{Flowing}$$

to the TBox then this concept is not satisfiable with respect to the TBox since the set of things which are rivers are defined to be flowing and thus no individuals can exist that are rivers but do not flow.

The other key inference subsumption is used to calculate the hierarchy of concepts according to their generality. Additionally all of the other inferences can be reduced to concept subsumption

$$\begin{aligned}
\text{C is unsatisfiable} &\Leftrightarrow \text{C is subsumed by } \bot \\
\text{C and D are equivalent} &\Leftrightarrow \text{C is subsumed by D and D is subsumed by C} \\
\text{C and D are disjoint} &\Leftrightarrow \text{C} \sqcap \text{D is subsumed by } \bot
\end{aligned}$$

In addition if a Description Logics language provides full negation of concepts then all inferences can be reduced to unsatisfiability

$$\begin{aligned}
\text{C is subsumed by D} &\Leftrightarrow \text{C} \sqcap \text{D is unsatisfiable} \\
\text{C and D are equivalent} &\Leftrightarrow \text{both (C} \sqcap \neg\text{D) and (}\neg\text{C} \sqcap \text{D) are unsatisfiable} \\
\text{C and D aredisjointg} &\Leftrightarrow \text{C} \sqcap \text{D is unsatisfiable}
\end{aligned}$$

The advantage of having full negation in a Description Logics language is that the reasoning on these languages can be performed by so called tableau-based algorithms [SchmSmol91]. To use these all inferences have to be reduced to unsatisfiability and thentableauu-based algorithms can be used to solve them. Since most modern Description Logics languages provide this construct a lot of research has gone into developing efficient algorithms for tableau-based reasoning [BaaderEtAl94], [Horr98], [HorrPate99] and [HaarMöll01a].

2.1.4 Open world reasoning

One important aspect to keep in mind when dealing with reasoning in Description Logics languages is that they all follow an open world assumption. This is especially important since knowledge representation systems bear a superficial similarity with database systems. The TBox is similar to the database schema and the ABox similar to the data in the database. The important difference is that databases use closed-world assumptions when answering queries. In a database if there is no individual that fulfils the query criteria then the assumption is that such an individual does not exist and that the statement that no such individual exists is true. On the other hand in the open-world reasoning of Description Logics if no individual fulfils the query criteria then the implication is that there is a lack of information. It is not possible to deduce that since the query is not fulfilled the negation of the query is true. Absence of proof is not proof of absence. This has to be taken into account when querying Description Logics knowledge bases.

2.1.5 Expressive Description Logics

The Description Logics of the AL family, while good for explanation purposes is ill-suited for real world applications since it lacks a number of constructs that are useful in real-world applications. To this end the S family of languages will be presented. The S languages are created from ALC by addition of transitively closed primitive roles [Satt96] and derives its name from its relationships with the propositional (multi) modal logic $S4(m)$ [Schi91]. This language can be extended with the following capabilities [HorrEtAl00a]

- Inverse roles symbolised by the letter I,
- A role hierarchy symbolised by the letter H,
- Qualified number restrictions on roles symbolised by the letter Q [HollBaad91],
- Unqualified number restrictions on roles symbolised by the letter N,
- Unqualified number restrictions that are functional restrictions of the form $\leq 1R$ and $\geq 2R$ symbolised by the letter F.

Adding these capabilities to a Description Logics language massively increases the complexity of reasoning. The two Description Logics that we will be interested in (SHOIN and SHIF) have a worst-case complexity of non-deterministic exponential time (NExpTime) and deterministic exponential time (ExpTime) [Tobi01] respectively. Luckily implementations of algorithms (FaCT [Horr98], RACER [HaarMöll01b], Pellet[ParsSivr04]) for these Description Logics have shown that for typical applications there are optimisations that can be applied [Horr98][HorrEtAl00b][HorrSatt02][HaarMöll00][HaarMöll01b] so that they can be employed inpractisee. Unfortunately there is currently no known practical algorithm for SHOIN that is complete. This means that the behaviour of reasoners for this language is less predictable and may lead to the answer *unknown* [HorrEtAl03].

2.2 Ontology Development

As stated earlier in this thesis the focus is on ontology languages that have their foundations in Description Logics and especially OWL. When developing ontologies in such languages a number of important development principles have to be taken into account. These can roughly be split into two areas. Development principles that arise from the underlying Description Logics system and principles that arise from the fact that a real world situation is to be modelled as correctly as possible.

2.2.1 OWL

OWL is the Ontology Web Language and is the W3C[1] recommendation for an ontology language to be used in the Semantic Web [BernEtAl01]. OWL is a relatively new ontology language (2001) but has evolved out of numerous previous languages and is therefore nevertheless a mature language. As such OWL has its roots in three areas [HorrEtAl03]

[1]World Wide Web Consortium, http://www.w3c.org

- *Description Logics*: The formal semantics of constructors and language features are derived from their semantics in Description Logics.
- *Frame Systems*: OWL provides a frame based view into the knowledge base to simplify understanding and working with OWL ontologies for people who are not experts in the field of Description Logics.
- *RDF*: OWL is designed to be an compatible with RDF (Resource Description Framework [2]. Thus every RDF graph is a valid OWL ontology.

In order to reconcile all these different influences and requirements OWL has been designed with three sub-languages that focus on different aspects of the requirements

- *OWL Lite*: OWL Lite is a subset of OWL DL and corresponds to the DL SHIF(D) (The D meaning that it is restricted to a domain). OWL Lite provides efficient reasoning but slightly restricts what can be expressed. Actually almost everything from OWL DL can be expressed except number restrictions with a cardinality not equal to 1 [HorrEtAl03].
- *OWL DL*: OWL DL provides a very expressive ontology language, while still being decidable and is very close to SHOIN(D).
- *OWL Full*: OWL Full is extremely expressive but inferences on OWL Full ontologies are not decidable any more. OWL Full is primarily necessary to retain compatibility with RDF graphs. Most automated reasoners will stop processing an OWL ontology if they detect that it belongs to OWL Full.

The ontologies presented in this thesis will either be in OWL Lite or OWL DL.

2.2.2 Basic Modelling Principles

As stated earlier there are two groups of development principles. First we will look at those principles that arise from the structure of the language.

Here one of the main problems that occur is the incorrect use of the $\forall$ and $\exists$ constructs. These differ in their interpretation in OWL from the interpretation that they have in natural language

- While in natural language the $\forall$ constructor is interpreted as meaning *each and every* in the Description Logics interpretation it has the meaning of *each and every if there is one.* Thus none is also a valid answer.
- The $\exists$ constructor is less problematic since it is equivalent to the natural language construct *at least one.* Nevertheless care is to be taken that the natural language assumption of *basically all* is not assumed to hold true in Description Logics as well.

This in addition to the fact that OWL uses open-world reasoning is the basis of one of the more common problems when modelling in OWL. The problem usually surfaces when calculating the subsumption hierarchy. Since open-world reasoning is used the reasoner will not created certain hierarchies even though to the casual viewer the concepts should be arranged in a hierarchy. For example given the two concepts and their definitions

[2]http://www.w3.org/TR/rdf-syntax-grammar/

$$\begin{aligned}\text{Coniferous Forest} &\equiv \forall\ \text{hasTree.ConiferousTree}\\ \text{Pine Forest} &\equiv \exists\ \text{hasTree.PineTree}\end{aligned}$$

the sumbsumption reasoner will not place the PINE FOREST as a subclass of the CONIFEROUS FOREST even though they seem to belong in a hierarchy together. The problem is that the definition for the CONIFEROUS FOREST says that all trees (if there are any) must be coniferous trees. The PINE FOREST definition on the other hand says that at least one of its trees must be a pine tree. Due to the way the $\exists$ constructor works no restriction is placed on what other trees may be in a PINE FOREST. These other trees may not be coniferous trees and thus the definition of the CONIFEROUS FOREST would be broken. Thus the reasoner will not create the hierarchy as desired. To create the hierarchy the definition for PINE FOREST has to be augmented as follows

$$\text{Pine Forest} \equiv (\exists\ \text{hasTree.PineTree}) \wedge (\forall\ \text{hasTree.PineTree})$$

This definition guarantees that the PINE FOREST will contain only pine trees which are coniferous trees and thus the desired hierarchy can be inferred.

It is thus important to make sure that the concepts keep the desired meanings when translated into Description Logics definitions.

2.2.3 Advanced Modelling Principles

In addition to the principles and problems outlined in the last section there are principles that should be followed to create a model of the domain of interest that is as correct as possible. At the same time the ontology should remain clear and easily understandable. Also to be taken into account is that knowledge implicit in concept and restriction names cannot be understood by reasoners and therefor as much knowledge as possible should be made explicit. These goals influence each other. For example adding knowledge to create a better model of the domain decreases the understandability. On the other hand increasing readability might make it impossible to describe certain aspects of the domain.

A further problem is that the language used and its capabilities influence the way that the knowledge is structured as found by [RussEtAl99]. They found that ontologies modelled in languages that do not directly support a reasoning system tend to be structured in a way that resembles a specification. On the other hand ontologies modelled in languages that are in some way tied to a reasoning system are structured in such a way as to make maximum use of the reasoning system.

To avoid these pitfalls it is useful to have guiding principles on how to insert knowledge into the ontology and how to structure this knowledge in the ontology. One of the most useful approaches to this problem is to make sure that the amount of explicit knowledge is maximised and that as little knowledge is implicit within the concept names or their hierarchy. This increase in explicitness improves both the model of the domain and also the readability of the ontology. The model is improved since an increase in the amount of explicit knowledge means that the domain is modelled more correctly and also that the modelled knowledge can be checked for correctness more easily. The readability improvement is not so obvious since explicit ontologies can be very verbose. Nevertheless readability is improved because this verboseness can be hidden by the modelling tool and more concise descriptions can be generated based on the explicit knowledge. Additionally the more explicit knowledge the better for reasoning, thus maximising the amount of explicit knowledge is very important.

A very useful framework in this area is an ontology refactoring approach described by [Rect03]. He applies his refactoring methodology to existing ontologies to increase the amount of explicit knowledge in the resulting refactored ontologies. If this refactoring method is used as a basic guideline when developing the ontology the result is an ontology that is very explicit while being easily readable.

The approach splits the ontology into two parts. The primitive skeleton and the defined concepts. The primitive skeleton contains a set of tree structures that describe very basic concepts and their hierarchical structure. The concepts of the primitive skeleton form the basic building blocks on which the ontology (or ontologies) describing the actual knowledge can be constructed.

For the primitive skeleton certain criteria must hold in order to guarantee the correctness and explicitness of the ontology

1. The primitive skeleton should form a tree. This means that the each primitive concept must have exactly one primitive parent.
2. Each sub-tree of the primitive skeleton should be homogeneous and the branching principle should be subsumption. The criteria upon which the subsumption is based should be consistent within each sub-tree and become narrower the deeper the tree is.
3. The primitive skeleton should clearly distinguish between self-standing concepts and partitioning concepts. Self-standing concepts are most objects found in the physical and conceptual world such as trees, houses, uses and an open world view should be assumed. Partitioning concepts are concepts that divide a space into multiple parts - e.g. ELEVATION can be divided into ALPINE, SUB-ALPINE, GREENLAND AND WOODS, ..., SEA-LEVEL. Partitioning concepts can be treated with a closed world assumption. Most concepts should be modelled as self-standing concepts, because only in very few cases can it be said that the list of concepts fully covers the space that the parent concept covers.

Figure 2.2 gives an example showing the part of a primitive skeleton describing trees. It observes the three rules given above since the branching principle is always subsumption and the branching criteria is always the type of tree. The concepts are self-standing and form a tree.

Special care has to be taken when creating the the primitive skeleton to ensure that the represented knowledge is precisely the knowledge that should be represented. Since the main ontology builds on the primitive skeleton any errors in the primitive skeleton have

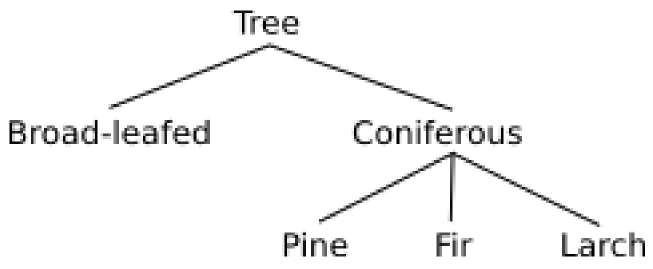


Figure 2.2: Example of part of a primitive skeleton describing trees

$$\begin{aligned}
\text{Broad Leafed Forest} &\equiv \text{Vegetation} \wedge \exists\ \text{hasTree.Broad-leafed} \\
\text{Coniferous Forest} &\equiv \text{Vegetation} \wedge \exists\ \text{hasTree.Coniferous} \\
\text{Pine Forest} &\equiv \text{Vegetation} \wedge \exists\ \text{hasTree.Pine}
\end{aligned}$$

Figure 2.3: Example of how defined concepts are constructed from the primitive skeleton

far reaching effects and are also hard to correct since in some cases major parts of the main ontology depend on these erroneous concepts.

For each of the main trees in the primitive skeleton at least one property is defined that has as its range that part of the primitive skeleton. These properties are then used in the definition of the actual knowledge that is to be represented in the ontology.

Finally using the properties defined on the primitive skeleton the knowledge that is to be represented in the ontology can be defined. This is done using necessary and sufficient conditions that restrict the properties to certain values. Figure 2.3 shows how the concepts from the primitive skeleton can be used to create the defined concepts.

These defined concepts do not have to be arranged in any kind of hierarchy. This is because based on the necessary and sufficient conditions a reasoner can infer the hierarchy of the defined concepts form. This frees the ontology developer from a further task, namely making sure that the hierarchy is still correct and consistent. In the example given above the reasoner can infer that since a PINE is a CONIFEROUS TREE then a PINE FOREST is a CONIFEROUS FOREST and thus the PINE FOREST can be placed below the CONIFEROUS FOREST in the hierarchy.

2.2.4 Ontology Modularisation

Up to this point the primitive skeleton and the defined concepts have coexisted in one ontology. While this makes development and change of the ontology easier it presents certain hurdles when more than one ontology is to be defined. In this case some kind of modularisation of the ontologies is necessary.

The cleanest and also most useful boundary for modularisation is splitting the primitive skeleton off from the main ontology. This creates one ontology that contains only primitive concepts called the skeleton ontology. The skeleton ontology can then bere-importedd into the ontology containing the defined concepts. It can now of course also be imported into multiple ontologies to create a set of ontologies that reference a common vocabulary and will later form the basis for the ontology integration.

This approach is unfortunately not free of problems and the main problem is that it makes maintenance work more difficult. Unfortunately while there is good tool support for editing OWL ontologies with the Protégé[3] ontology editor, there is currently no support for modifying such modularised ontologies. This is not problematic when adding or moving concepts in the skeleton ontology, but removing or renaming concepts is difficult. Removing primitive concepts is slightly easier than renaming them since all that has to be done is to remove all definitions from the defined concepts that use the primitive concept that is to be removed. Then the primitive concept can be removed. Renaming a concept is much more difficult since if the primitive concept is simply renamed then loading the ontologies with the defined concepts becomes impossible since the editor tries to load the primitive concept under the old name under which it no longer exists and which causes the loading to fail. For this reason renaming a concept has to be split into adding a new primitive concept under the new name, then changing all definitions from the old to the new primitive concept and finally removing the old primitive concept. This process can be improved using scripts but the assumption is that the ontology design is done by domain experts who need proper tool support. This just illustrates how important it is that proper care is taken when designing the skeleton ontology to minimise the amount of change that is necessary.

[3]Protégé Ontology Editor, http://protege.stanford.edu

3 Existing Matchmaking Algorithms

Since heterogeneous data sources are a reality in many areas a lot of matchmaking algorithms have been proposed. To gain an overview over the existing algorithms it is useful to split them into certain groups that show similar behaviour. The most defining criteria is whether the algorithms are [RahmBern01]

- *Schema based* matchmaking algorithms only make use of the information available at the schema level (names, descriptions, constraints...) or
- *Instance based* matchmaking algorithms use meta-data on the instances and statistics collected from the actual data for the matching process.

In addition to those two basic categories the following further orthogonal criteria can be used to classify matchmaking algorithms [MadhEtAl01]

- *Element* or *Structure* granularity describes whether the algorithm works on elements such as table columns for a schema based algorithm or only larger structures such as a table row in an instance based algorithm.
- *Linguistic based* matchmakers use the names of schema elements or other textual descriptions and apply linguistic functions such as stemming or tokenising to find matches between words. Additionally linguistic information such as synonyms, homonyms, hypernyms, domain-thesauri are used possibly in combination with information retrieval techniques.
- *Constraint based* algorithms uses schema constraints such as data types, value ranges or uniqueness in the matchmaking calculation,
- *Matching cardinality* specifies whether the matchmaker will only return 1:1 mappings between the compared elements or whether other mappings such as 1:n or n:1 are also supported.
- *Auxiliary information* is used in some matchmakers to augment the information derived from the schema or instances to improve the quality of the mappings. This information can be human input or results from previous matchmaking runs.
- *Individual*, *Multiple* or *Hybrid* approaches can be taken to matchmaking. The individual algorithms obviously use only one criteria to calculate the mappings. Multiple matchmakers use multiple criteria independently and then aggregate the results while hybrid matchmakers combine multiple criteria when creating the mappings.

In practise it is very uncommon that one individual criteria is sufficient to produce sensible and useful mappings between elements. Thus multiple or hybrid matchmakers are the norm. Additionally most matchmaking algorithms focus on the schema level since it is easier to work with and contains cleaner information that can be handled more easily by the matchmaker.

3.1 Schema Level Matchmaking

Schema based matchmaking is much more common than instance based matchmaking. This is probably due to the fact that much work in this area has been done by people working in the field of databases where schema integration is already a well known problem. Five groups of schema based matchmakers will be analysed in detail

- *Lexical matching* compares schema element names,
- *Dictionary or WordNet* matching uses a dictionary, most commonly WordNet to match elements,
- *Structural matching* uses the schema structure for mapping information,
- *Hybrid matchmakers* combine the previous algorithms to improve the quality of the resulting mapping,
- *Description Logics* matchmaking is based on Description Logics reasoning.

3.1.1 Lexical Matchmaking

Lexical based matchmaking algorithms work on the names of the elements to compare. Of these algorithms the most frequent is word or edit distance [WillEtAl03] [MaedStaa02b] [SycaEtAl99] [MorkBern04]. The basic function ED calculates the number of deletions, additions and substitutions are necessary to transform the first word to compare into the second one. So for the words FOREST and FORESTS the ED(FOREST, FORESTS) = 1 since only one letter (the final S) needs to be added.

This edit distance is then incorporated into a weighted formula that weighs the edit distance against the length of the shorter of the two words to calculate the final similarity measure

$$similarity(L_1, L_2) = max(0, \frac{min(|L_1|, |L_2|) - ED(L_1, L_2))}{(min(|L_1|, |L_2|))})$$

This results in a similarity measure between 0 and 1 where 0 is a bad match and 1 is a complete match. 0 is a bad match but not necessarily a complete mismatch. This is due to the fact that the similarity comparison is cut off by the maximum operation at 0 even if it would be a negative value. Thus while similarity measures with values above 0 can be compared those with a value of 0 must be ignored since one value might just be a little less than 0 while the other might actually be a lot less than 0.

In our example from above the two words FOREST and FORESTS the complete similarity measure would be

$$\mathit{similarity}(\text{Forest}, \text{Forests}) = \mathit{max}(0, \frac{(6-1)}{6}) = \frac{5}{6}$$

While this similarity measure is easy to calculate and seems to offer decent results, there are a number of problems inherent in this method.

The first and biggest problem is that this similarity measure completely ignores the semantics of the two words. That the word FORESTS is a plural of the word FOREST is ignored as well as the fact that a FOREST is an area that contains trees. Although it exhibits this basic flaw it is nevertheless used as the basis in different matchmaking algorithms [WillEtAl03][MaedStaa02b][SycaEtAl99][MorkBern04][MaedStaa02a]. In addition to this basic problem there are additional problems that are related to the way the similarity measure is calculated but they are all grounded in the fact that the semantics of the words are ignored.

The first is that the different word pairs with the same similarity measure may actually not have the same similarity if evaluated by a human.

$$\begin{aligned}
\mathit{similarity}(\text{Bird}, \text{Bard}) &= \frac{3}{4} \\
\mathit{similarity}(\text{have}, \text{shave}) &= \frac{3}{4} \\
\mathit{similarity}(\text{have}, \text{gave}) &= \frac{3}{4}
\end{aligned}$$

All three pairs have the same similarity measure but their similarities are obviously not the same. The first difference is that the first pair contains two nouns, the second pair contains a verb and a word that can be both a verb and a noun and finally two verbs. If we ignore this then there is still the problem that the BIRD and the BARD have little in common except singing and possibly a colourful exterior, HAVE and SHAVE have nothing in common while HAVE and GAVE have something in common since you have to HAVE something to HAVE GIVEN it to someone (although even here the verb forms don't really work). To round out the problems in this example there is the word shave which can be both a noun and a verb. In this similarity measure both forms are just as similar although in the real world the similarity depends on the context. If you look at the grammatical functions they are either both verbs or a verb and a noun.

Unfortunately this problem also works in reverse. Things that to a human are very similar may get similarity measures of 0 as shown in the following example

$$\begin{aligned}
\mathit{similarity}(\text{Forest}, \text{Wood}) &= 0 \\
\mathit{similarity}(\text{Forest}, \text{Bosk}) &= 0 \\
\mathit{similarity}(\text{Wood}, \text{Bosk}) &= \frac{1}{4}
\end{aligned}$$

FOREST and WOOD get a similarity measure of 0 meaning a bad match even though they actually mean almost the same thing. A similar problem plagues FOREST and BOSK[1] and although they are not synonyms they are in an is-a relationship and should be more similar than 0. At the same time the similarity for WOOD and BOSK is $\frac{1}{4}$ so although for the FOREST the similarity to BOSK is 0, for the synonym of FOREST namely WOOD the similarity to BOSK is $\frac{1}{4}$. While sometimes a synonym is closer to another word than the original word, if the difference in similarity is $\frac{1}{4}$ then this is problematic.

In addition to all these problems the basic assumption underlying the edit distance similarity measure is that both words are in the same language.

$$similarity(\text{Forest}, \text{Wald}) = 0$$

Here WALD is the German word for FOREST so actually they are the same and should have a similarity value of 1 not of 0.

The only real advantage that this approach offers is that it does not require a predefined shared vocabulary but can be applied to arbitrary schemas. Unfortunately due to the problems that are inherent in this method the results from comparing arbitrary schemas are actually not semantically significant. Nevertheless algorithms based on word distance are still used and often form the basis for structural or hybrid matchmaking.

The quality of the results can be improved by first applying linguistic functions to the words such as stemming. This improves such problems as noun / verb distinguishing. Nevertheless it does not solve the basic problem that the results of the similarity measure are not semantically meaningful in any way.

3.1.2 Dictionary or WordNet Matchmaking

As shown in the previous section it is basically impossible to provide a sensible measure for how similar two concepts are without some common base to start from. This implies that for any kind of matchmaking some kind of shared vocabulary is required which can be used as a reference when comparing concepts. One of the oldest forms of shared vocabularies are dictionaries. Dictionaries contain lists of words, their definitions and certain relations between the words. The only problem is that these dictionaries are usually not in a form that the computer can process.

This shortcoming is addressed by WordNet[2]. WordNet is an on-line lexical database for the English language [Miller95]. It follows the traditional principle of categorising words into the four basic syntactic categories nouns, verbs, adjectives and adverbs. The words are interrelated via six semantic relations

- *Synonym* is the most fundamental relation in WordNet. The synonym relation is a symmetric relation between two words meaning that both words have the same meaning. All words in WordNet are organised into sets of synonyms (called synsets) which form the basic structure of WordNet.

[1]Merriam-Webster On-line Dictionary: A small wooded area
[2]WordNet is a registered trademark of Princeton University

FOREST: a dense growth of trees and underbrush covering a large tract.
WOOD: A dense growth of trees usually greater in extent than a grove and smaller than a forest

Figure 3.1: Definitions of FOREST and WOOD taken from the Merriam Webster On-line Dictionary

- *Hyponym* and *Hypernym* are the secondary organising principle. If a word is the hyponym of another word then the first word has a narrower definition that the second. Inversely the second word is the hypernym of the first word. They are both transitive relations and are their respective inverse relations.
- *Meronym* and *Holonym* form another pair. A meronym is a concept that is part of another concept. The holonym is the inverse. These two relations are again transitive.
- *Antonym* is the opposite concept. Similar to the synonym relation it is symmetric and is mostly used in the organisation of adjectives and adverbs.
- *Troponym* is the equivalent of hyponym for verbs.
- *Entailment* relations between verbs mean that for the activity of one verb to occur the the activity of the entailed verb also has to have happened.

These relations among the words then form the basis for the similarity measurement between concepts [WillEtAl03][CastEtAl04][CastEtAl03][CastEtAl05][MaedEtAl02]. The similarity measure is usually calculated as the number of relations that have to be traversed from the one concept to the other. In addition each relation type is given a certain weight and these weights specify the similarity of the two concepts that the relation connects. For example two words connected by a synonym relation might have a similarity weight of 1 (meaning they are equal) whereas two words connected via hyperonymy might have a similarity weight of 0.8 (they are similar but there is a certain difference somewhere between them) [CastEtAl03]. These weights are then combined to calculate the total similarity between two concepts.

Unfortunately there are also problems inherent in the use of WordNet and its relations for similarity measurement. The main problem is the definition and usage of synsets. The following example shows the synset of the word FOREST

FOREST has synonyms: WOOD, WOODS

The problem is that while WOOD might be a synonym of FOREST the fact that they do not mean exactly the same as is clear from the definition taken from the Merriam Webster On-line Dictionary[3] (fig. 3.1). The definition of the word WOOD is very clear since it defines a WOOD as being a growth of trees that is not as large as a FOREST. Thus they are actually not synonyms since they don't really describe the exactly same concepts.

In addition the word WOODS is elevated to the same level as the singular WOOD. This means that there is basically no difference between the words WOOD and WOODS. Obviously this is not entirely correct since WOODS denotes an area with trees that is much larger than the one denoted by WOOD. This difference in definition is totally lost when

[3]http://www.m-w.com

Word	**Hyponyms**	**Meronyms**
Forest	Bosk, Grove	Undergrowth, Trees
House	Cabin, Chalet, Villa	Library, Porch

Table 3.1: Hyponyms and Meronyms of FOREST and HOUSE taken from WordNet

the two words are put into the same synset. While this difference might seem distant and constructed assume that one data set uses the word FOREST while a second one differentiates between WOOD and WOODS. Which of the two words should FOREST be assigned to? Based on the synset information it doesn't matter since they are all the same, but based on the definition given above in the dictionary it makes more sense to assign FOREST to WOODS than WOOD since their definitions have more overlap.

A further problem is that the principles based on which hypernym/hyponym and also meronym/holonym are determined is different for each concept. In the example given in table 3.1 the hyponyms of FOREST are BOSK and GROVE. Already here although BOSK and GROVE are on the same relations level the reason why they are hyponyms of FOREST are different. A BOSK is a small forest that is very dense and dark. On the other hand a GROVE is a type of forest that lies in a depression and thus has a very specific shape. Again the qualitative differences between the two words is lost when they are grouped together in the set of hyponyms.

In addition to the branching principle being inconsistent within one word the branching principle is also not consistent for different words. So while the branching principle for FOREST is the size of the forest and the shape of the ground on which the forest stands, the principle for HOUSE is the form and elaborateness of the house. The result of this inconsistency is that when traversing the structure from one concept to the other since the branching principles are different the results of the similarity measurement become unreliable.

Furthermore in this example the words that are meronyms do not have the same quality. UNDERGROWTH and TREES reference groups of real world objects, the word LIBRARY refers to a specific room in the house where there are books and somewhere to read. So while UNDERGROWTH and TREES are sets of real world objects, the LIBRARY is defined via the use that it can be put to, but these differences are not visible, when just looking at the relations in WordNet.

The final problem is that the WordNet dictionary is not domain specific. While this may seem like an advantaged at first since it can easily be plugged into a similarity measurement for any domain, the actual result is that it is often to vague and imprecise for any use. Often in specific domains words receive new or slightly modified definitions that hold only within that domain. This domain specific knowledge would need to be grafted onto the WordNet system thus defeating the purpose of the WordNet system.

3.1.3 Structural Matchmaking

A totally different approach to similarity measurement is structural matchmaking. Structural matchmaking uses the structure defined in the ontologies to calculate the

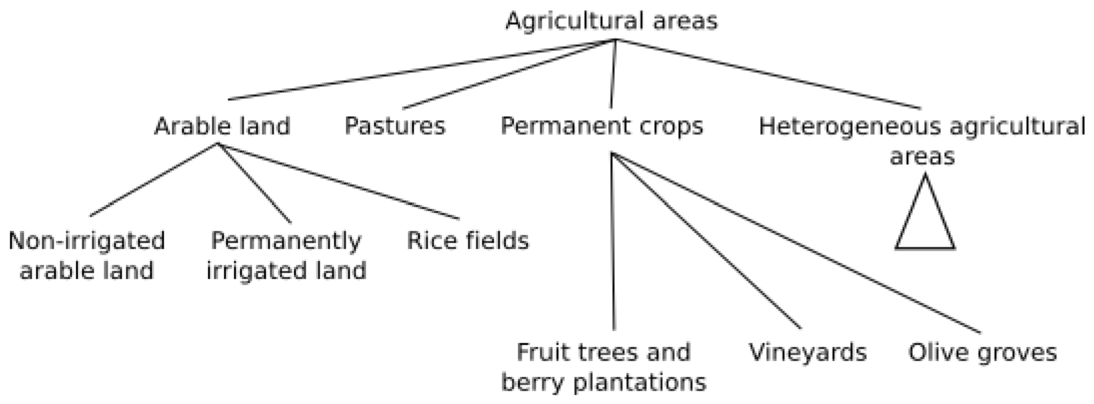

Agricultural areas
Arable land
Pastures
Permanent crops
Heterogeneous agricultural areas
Non-irrigated arable land
Permanently irrigated land
Rice fields
Fruit trees and berry plantations
Vineyards
Olive groves

Figure 3.2: Structure of the Agricultural Areas in the Corine catalogue

similarity between concepts. The idea behind it is that similar concepts will also have similar set of surrounding concepts. In its simplest variety the similarity is based on the number of children, parents and properties that the two concepts have [MaedStaa02b][MorkBern04][CastEtAl04][CastEtAl03][CastEtAl05][MaedStaa02a].

Unfortunately the basic assumption that similar concepts will have similar surrounding concepts is based on the faulty assumption that the same domain modelled by different people or groups will nevertheless result in a similar structure. While this may hold for very simple domains it immediately fails when looking at a more complex domain such as land-use. For example when comparing the European Corine land-use classification to the Austrian Realraumanalyse there are massive differences between the structure in some areas. Figures 3.2 and 3.3 show the structures of agricultural areas of the Corine and Realraumanalyse land-use catalogues.

As you can see the Corine catalogue has a very strongly hierarchical approach with three levels and only the third level containing those categories that are actually encoded in the data. On the other hand the Realraumanalyse catalogue uses a matrix style representation with one axis containing the different inclinations of the ground and the other axis containing the actual usage categories. These structures can not be compared directly and the only way to make them comparable is to transform the Realraumanalyse structure into a list. After such a transformation the Realraumanalyse has a very flat two level structure when compared to the three level Corine structure. A structural comparison between the two root concepts would result in a very small similarity value even though the two concepts are exactly equal.

While the difference in the conceptualisation between the two classifications is not so strong in the remaining parts of the classifications it is still strong enough to make the structural comparison very imprecise. The Realraumanalyse is flatter and the different hierarchies are not as cleanly modelled as in the Corine classification. For example the Realraumanalyse contains a section called Other areas which acts as a kind of odd categories collection box containing everything from rivers and wetlands to areas belonging to the public administration. These kinds of unclean structures make structural comparisons very imprecise.

The final problem is that while it may seem that the Corine and the Realraumanalyse

Inclination of the agricultural areas

- Flat
- Slight inclination
- Moderate inclination
- Steep
- Mountain pasture elevation

Primary agricultural use

- Arable land > 90%
- Arable land dominates with pastures > 10%
- Arable land and pastures mix 40 - 60 %
- Pastures dominate with arable land > 10%
- Pastures > 90%
- Vineyards, arable land - vineyard complexes
- Specialised crops with arable land - pasture areas
- Wetlands (moors in agricultural areas)
- Pastures that are also winter sport areas

Figure 3.3: Classification of the Agricultural Areas in the Realraumanalyse catalogue

classifications describe the same domain, that is actually not true. The Corine classification is designed to cover all of Europe. The Realraumanalyse classification on the other hand is only designed for Austria. This results in such things as that the Realraumanalyse does not contain categories for seas and beaches. At the same time it contains very detailed classes such as Kellergassen which are roads in wine growing areas that are dotted with wine cellars. In addition to these differences that come from the slightly varying domains other structural differences come from the fact that Corine data is only generated from satellite images while the Realraumanalyse incorporates data from other sources such as zoning plans and historical information as well. This leads to a strongly varying granularity of the classifications with Corine encompassing 64 categories and the Realraumanalyse a much larger 136.

All these differences starting from different modelling principles over slightly different domains to the granularity of the classifications make structural matchmaking very imprecise. Add to this the fact that matching based on the structure is not semantical, since the meanings of the concepts to be compared is completely ignored strongly reduces the utility of structural matchmaking.

3.1.4 Hybrid Matchmaking

Using a single algorithm for similarity calculation and matchmaking reduces the stability of the matchmaking result. Therefore a frequent approach is to combine the results of two or more matchmaking algorithms to create a hybrid and more robust matchmaking algorithm. Usually these are combinations of WordNet and structural algorithms [CastEtAl04][CastEtAl03][CastEtAl05] or lexical and structural algorithms

[MaedStaa02b][MorkBern04][MaedStaa02a], but also WordNet and lexical [WillEtAl03].

The results of the different algorithms can be combined in two ways. Either the results from the different algorithms are combined after they have been calculated separately [WillEtAl03] [MaedStaa02b] [CastEtAl04] [CastEtAl03] [CastEtAl05] [MaedStaa02a] or the results of one algorithm are integrated into the other algorithm [MorkBern04] [MaedEtAl02]. Both approaches have their advantages and disadvantages

- *Separate calculation.* The advantage of performing separate calculations for each of the algorithms to be combined is that the algorithms can be developed and tuned independently. Also incorrect results from one of the algorithms don't have such a strong influence on the final result. For example if a lexical and a structural algorithm are combined and the lexical algorithm gives a false positive then this can to a certain extent be corrected by the structural algorithm. The disadvantage is that each algorithm starts with zero knowledge about the possible relations between the different concepts to be matched. They cannot make use of the results that have been derived by the other algorithms. Assume that in the one hierarchy there is a parent concept with one child and in the other hierarchy there is a parent concept with three children. If all of the three children in the one hierarchy have been found to be very close to the one child in the other hierarchy by a WordNet algorithm then this information cannot be used by the structural algorithm. The structural algorithm will have to decide that the structures are not too similar, even though since the three concepts are all very similar to the one concept then the hierarchies are actually very similar.
- *Integrated calculation.* The advantage of using an integrated calculation approach is that the results from the first algorithm can be used in subsequent algorithms to improve the quality of their results. In a WordNet and structure algorithm combination if two concepts have a very high similarity value in WordNet then this information can be used by the structural algorithm to see whether the the hierarchies are also similar. The problem with the integrated approach is that it propagates false positives and false negatives. If for example the result of a WordNet matching between two concepts incorrectly returns that they are completely dissimilar then this result will also mean that concepts that are near them in the hierarchy get lower similarity results. The reason for this is that the structural reasoner knows that these two concepts are completely dissimilar and therefore assumes that it is highly unlikely that the surrounding structure and its concepts is as similar as they possibly are.

Although hybrid similarity reasoners tend to achieve improved similarity results the basic problem remains that since they do not use algorithms that provide a semantically valid result, they themselves are not semantically valid. This problem cannot be resolved just by adding more similarity reasoners.

3.1.5 Description Logics Matchmaking

The last major approach to schema level matchmaking that will be looked at is the use of Description Logics in matchmaking. This approach is especially favoured when the concepts to be matched are encoded in ontology languages that are based on Description Logics. In this case the necessary definitions are already available and automated reasoning can easily be used. Using Description Logics based matchmaking algorithms means relying on a common vocabulary for the atomic terms and relations since equality or inequality of atomic terms and relations is defined via the comparison of their names.

The Description Logics matchmakers take the concepts and attempt to use subsumption to insert the concept from the one knowledge base into the other knowledge base. Depending on whether and if then where the concept from the one knowledge base is inserted into the hierarchy of the other knowledge base different levels of match are distinguished. The most basic distinction is between exact match and disjoint match. Apart from this distinction the following match classifications are used

- *Exact* match. The exact match means that the reasoner used to classify the concepts has determined that the concepts are equivalent from a Description Logics point of view. This is of course the best possible result from the comparison of two concepts. It is provided by all Description Logics matchmakers [KlienEtAl04][LiHorr04][LemmAren04][CastEtAl01][PaolEtAl02].
- *Plug-in* match. A plug-in match is given if the concept to be inserted into the hierarchy from the first ontology is subsumed by the concept in the second hierarchy. The two concepts are connected via an is-a relationship and while not completely exact are assumed to be very similar. This match class is always provided along with the subsume match [LiHorr04] [LemmAren04] [CastEtAl01] [PaolEtAl02].
- *Subsume* match. A subsume match is the opposite of a plug-in match. The concept to be inserted into the hierarchy subsumes the concept in the target hierarchy. This kind of match is ranked lower than the plug-in match although they are very similar. The reason for this is that the similarity between a concept and its super concept is seen as asymmetric. The sub-concept is more like the super-concept than the super-concept is like the sub-concept.
- *Intersection* match. The intersection match occurs when the two concepts cannot be arranged in a subsumption hierarchy, but do not conflict [LiHorr04] [LemmAren04] [CastEtAl01]. The inclusion of this match class is an improvement on algorithms that do not provide this match, because all concept pairs that cannot be put into a subsumption hierarchy are not immediately added to the disjoint or fail bucket. Thus the granularity of the results is improved.
- *Disjoint* match. A disjoint match means that parts or all of the definitions of both concepts conflict. It again is provided by most Description Logics matchmakers [KlienEtAl04] [LiHorr04] [LemmAren04] [CastEtAl01] [PaolEtAl02].

The basic problem with these algorithms is the rough discreet classification of matches. While the exact and disjoint match classes are cleanly defined and easy to interpret this does not hold for the other three classes.

For the *plug-in* match class the question is what to do if the concept to insert is subsumed by more than one concept in the target ontology. If these concepts in the target ontology subsume each other then the solution is simple as the source concept can be inserted below the most specific subsumer and the target concepts sorted by distance from the

$$
\begin{aligned}
\text{Public Parks} &= \text{hasSurface.GreenArea} \wedge \text{hasPrimaryUse.Leisure} \\
&\quad \wedge \text{ hasOwnership.Public} \\
\text{Green Areas} &= \text{hasSurface.GreenArea} \wedge \text{hasPrimaryUse.Leisure} \\
&\quad \wedge \text{ hasLocation.Urban} \\
\text{Airports} &= \text{hasSurface.Artificial} \wedge \text{hasPrimaryUse.Transportation} \\
&\quad \wedge \text{ hasBuilding.Airport}
\end{aligned}
$$

Figure 3.4: Description Logics definitions of PUBLIC PARKS (Realraumanalyse), GREEN AREAS (Corine) and AIRPORTS (Corine)

inserted concept. Unfortunately this does not solve the problem of how to order if the two target concepts that subsume the source concept are not arranged in some kind of is-a relation. In this case the algorithm can only assume that the two both have the same similarity to the source concept, even though the one concept might be very specific and almost the same as the source concept and the other more abstract.

The *subsume* match class suffers from the same problem as the *plug-in* match in that it is nearly impossible to say which of two subsume matches is better. In addition to this there is also the problem that although it is the exact opposite of the plug-in match it is ranked lower. This implies an asymmetric view of the concept hierarchies. The assumption is that the more specific concept is more similar to the more general concept than vice versa and this does not necessarily hold.

Finally there is one major problem with the *intersection* match and that is that it is basically the class into which everything is sorted that doesn't fit into the other classes. The algorithms assume that most things will fall somewhere into the subsumption hierarchy and if they don't and are not conflicting then they are probably not very similar nevertheless. Unfortunately most things do not form a subsumption hierarchy but exist next to each other. This is especially the case when the two concepts that are compared are on a similar level of abstraction and this is actually the more probable scenario than forming a subsumption hierarchy.

The problem is nicely illustrated when you compare the PUBLIC PARKS category from the Realraumanalyse to the GREEN AREAS of Corine (see fig. 3.4). Both have slightly differentiating properties. One is restricted to urban areas, the other is restricted to areas belonging to the public administration. Thus when compared they would be classified as an intersection match. The same is true if you compare the PUBLIC PARKS to the Corine AIRPORTS. They do not subsume each other, but they also do not conflict. From the point of view of the algorithm the similarity between the two pairs is the same, but it is obvious that the GREEN URBAN AREAS are closer to PUBLIC PARKS than to AIRPORTS.

There are also more differentiated and more granular algorithms based on Description Logics. These use subsumption not on complete concepts, but on parts of the definition [DiNoEtAl03][BenaEtAl05]. Thus the similarity results also become more fine grained and problems such as the one stated above can be partially solved. The problem that remains is that these algorithms only calculate the number of definition parts that can be

$$\begin{aligned}\text{Lake} &= \text{WaterArea} \wedge \forall\ \text{hasWater.}(\text{Standing} \wedge \text{Fresh}) \\ \text{InlandWater} &= \text{WaterArea} \wedge \forall\ \text{hasWater.Inland} \\ \text{Water} &= \text{WaterArea}\end{aligned}$$

Figure 3.5: Description Logics definitions for a partial matching algorithm

matched and those that can't and then from these values calculates the similarity value. How closely those parts that match match and how far apart the non-matching parts are is not taken into account. Additionally since Description Logics in some parts are not fully intuitive some at first strange results can occur.

Given the definitions in figure 3.5 the algorithm will find that LAKE and WATER are more similar than LAKE and INLAND WATER even though LAKE and INLAND WATER both have definitions that give information on the kind of water that they contain. The reason for this result is that although LAKE and INLAND WATER have similar definitions on their water types, the definitions conflict from a Description Logics point of view. At the same time LAKE and WATER are fully compatible since WATER = WATER AREA in Description Logics implies WATER = WATER AREA $\wedge$ $\forall$ HASWATER.$\top$.

Such results while fully correct are slightly non-intuitive at first which is why Description Logics matchmakers might be correct, but the results appear strange at first.

3.1.6 Other Matchmaking Approaches

In addition to these common matchmaking approaches there are also a few others that employ less common matching algorithms

- *Machine Learning.* In [DoanEtAl02] a system is described that uses a machine learning approach to calculate a set of constraints that the two ontologies to match and then applies relaxation labelling to create a mapping between the two ontologies that satisfies these constraints.
- *Rule based.* [TangEtAl03] describe a matchmaking system that uses a rule based approach. The ontologies and the matchmaking rules are stored in a deductive database with which the mapping can then be calculated
- *Approximation.* Using a heuristic function that approximates the actual similarity function [EuzeValt03] defines a system that iteratively approximates the similarity result. The heuristic is used to generate in initial mapping then the similarity function evaluates this and the results of the evaluation are fed back into the heuristic, thus iteratively approximating the similarity result.
- *Semi-automatic* and *manual.* In addition to these automatic matchmaking systems there are also those [NoyMuse00][MedjEtAl03] that are semi-automatic or manual in their mapping generation. A domain expert generates the mapping and the system can then apply it to the actual data.

The strengths and weaknesses of these algorithms will not be looked at in detail since that would exceed the scope of this thesis.

3.2 Instance Level Matchmaking

While schema level matchmaking only takes into account information available at the schema level, instance level matchmaking uses the information available at the data instance level such as property values or frequency of occurrence. Most of the matching principles given in the section on schema level matchmaking can also be applied to instance level matchmaking, especially such simple ones as lexical or dictionary matchmaking. They work in roughly the same way and thus share their advantages and disadvantages. For this reason only statistical matchmaking will be analysed since it can only be applied to instances.

Statistical matchmaking relies on the principle that the relative frequency of instances in the data sets will be similar for different data sources, since both data sources describe the same domain. It is not intended to be used as the sole matchmaking algorithm for instance integration, but in combination with other instance or schema level matchmaking algorithms could improve both the quality and the speed of the matchmaking process. The reason for this is that the matchmaker could ignore those concept pairs whose instance frequencies vary widely, thus improving speed and also precision since it reduces the chance of false positives.

This principle has been tested using data sets with land-use and land-cover datas describing Carinthia, Friuli Venezia-Giulia and Slovenia. Due to the nature of the tested datas the conclusions given here are only valid for land-use data sets and cannot readily be generalised.

Land-use data sets contain an unordered list of polygons with each polygon also having an associated list of attribute→value pairings. Amongst these attributes is a code attribute that identifies the land-use category the polygon belongs to. These land-use data sets for the three regions were analysed and polygon frequency and polygon area calculated for each land-use category.

A first look at the two statistics shows that there is little correlation between the polygon frequency and the polygon area. This is especially true for the Austrian Realraumanalyse dataset. For example while the most frequent category in the Realraumanalyse AREAS WITH MORE THAN 90% PASTURES ON STEEP HILLSIDES is also the second largest, the second most frequent category SPLITTERED SETTLEMENT AREAS ranks 21st in the area list. As can be seen in figure 3.6 while some classes more or less stay in the same order in both lists, there is too much movement to be able to say that there is a correlation between the two and that the frequency statistic could be used instead of the area statistic.

The same can be said for the Italian Moland dataset. On the other hand the Slovene Corine catalogue is the exception in that there is a certain correlation between the frequency and area. The reason for this is that the Corine catalogue employs a much larger scale when the data is acquired. This leads to larger polygons and thus the polygon ranking is more similar to the area ranking. Nevertheless since there is little correlation

Frequency ranking

1. Pastures > 90%, steep
2. Splittered settlement areas
3. Open development in general
4. Coniferous forest dominates
5. Mixed forest, broad-leafed dominates
6. Mixed forest, coniferous dominates

Area size ranking

1. Coniferous forest dominates
2. Pastures > 90%, steep
3. Mixed forest, broad-leafed dominates
4. Mixed forest, coniferous dominates
5. Alpine turf, partially with trees
6. Continuous alpine turf vegetation

Figure 3.6: Frequency and area size rankings for the Realraumanalyse (top 6 categories)

in the other two datasets and area is the defining attribute of land-use data from here on only area statistics will be taken into account.

Analysis of the area statistics shows that although the three datasets describe the same type of real-world situation and are also from geographically close areas, there are differences that arise from the geographic properties of the described area, the granularity of the land-use catalogue and also the methodical rules which were used in the data acquisition. For example in Carinthia the dominant (roughly 42% of the total area) land-use class is Coniferous forest dominstes while in Italy it is Non irrigated arable land.

The differences between the Carinthian data set and the Italian one are mainly down to the differences in the actual geography. Due to the topography and also industrial reasons Carinthia has a huge proportion of mainly coniferous forests that cover the hills and mountains. The settlements and agricultural areas are grouped together in the valleys and basins (fig. 3.7). On the other hand while the northern part of Friuli Venezia-Giulia is also hilly and mountainous it then quickly flattens out into a large plain which of course offers perfect conditions for farming and thus arable land dominates. It is has a belt structure (fig. 3.8). In the north it is mountainous with a structure similar to Carinthia. Then it becomes more hilly giving way to the broad-leaved forest belt and finally the agricultural belt. The situation in Slovenia is similar (fig. 3.9). It starts off with mountainous forests and towards the east becomes flatter and agriculture takes over. It is clear that even over the relatively short distances involved in the three regions the differences in topology and climate lead to very different land-use and land-cover patterns.

This problem is worsened by the fact that the land-use catalogues for Carinthia and Slovenia have a very different granularity. In the Carinthian Realraumanalyse about 15 different arable land categories are differentiated, while the Slovenian Corine catalogue only knows of one arable land category. Since there are so many different categories in the Realraumanalyse naturally each of the categories ranks lower in the hierarchy of land-use area than if they were aggregated into one category. Unfortunately to take advantage of

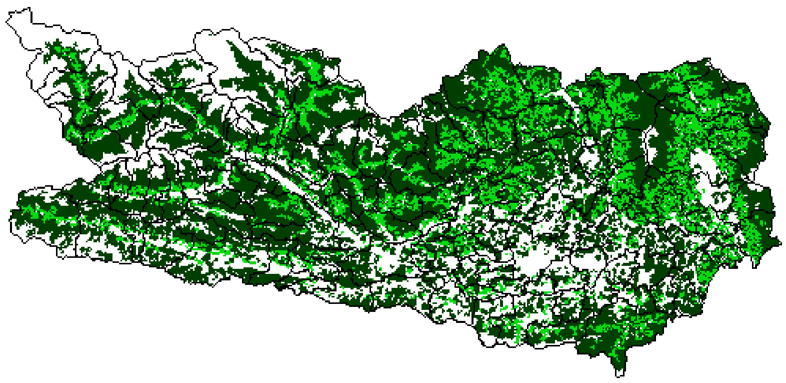


Figure 3.7: Structure of land-use in Carinthia (data from the Realraumanalyse). Clearly visible is the valley structure.

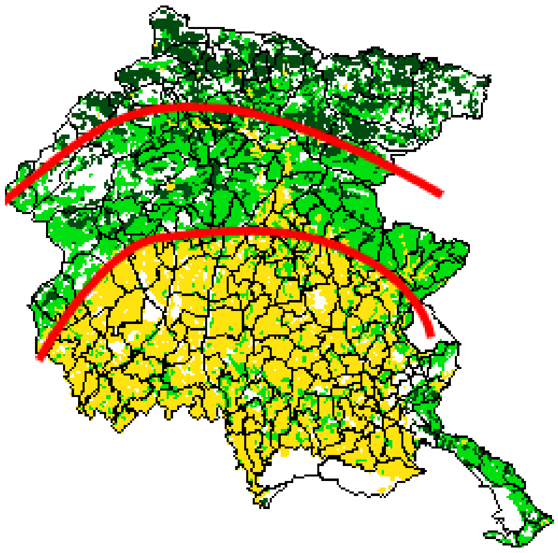


Figure 3.8: Structure of land-use in Friuli-Venezia Giulia (data from the Moland dataset). Shown are the three major land-use belts.

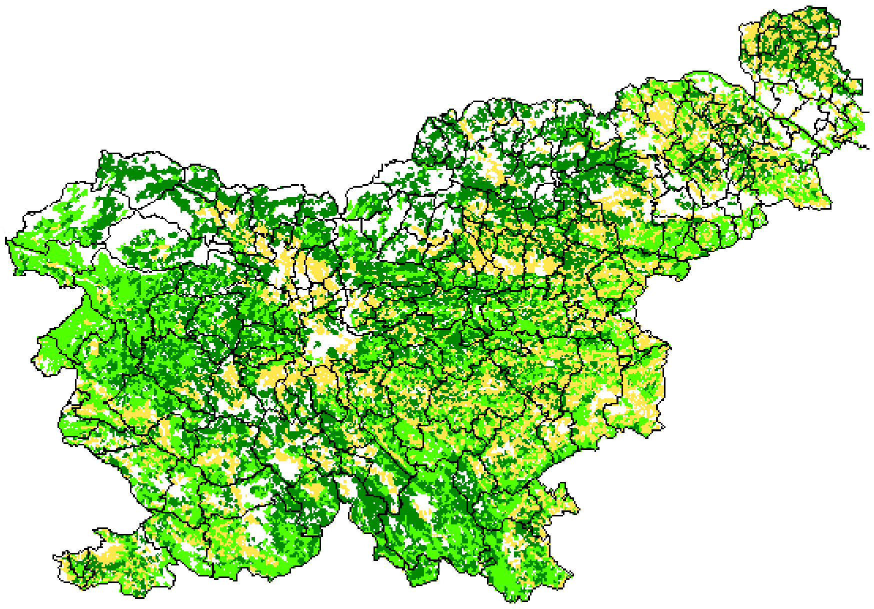


Figure 3.9: Structure of land-use in Slovenia (data from Corine). Structures are not so clearly visible due to imprecise data.

this knowledge would mean knowing when to compare base categories and when aggregated categories and this would defeat the purpose of fully automated harmonisation of data.

Finally the ranking is also influenced by the fact that the land-use datas were acquired with different methodologies. This once again is especially visible with the Corine catalogue. Corine for example discards all settlement areas with an area of less than 25 hectare. This of course removes small villages with only a few houses from the data set.

All these factors together lead to the conclusion that the ranking of land-use categories based on either frequency or area cannot be used for harmonisation purposes. Since the rankings are also not similar when the areas are aggregated the statistics can also not be used to improve the speed of the harmonisation algorithm.

4 Semantic Matchmaking Algorithm

All the matchmaking algorithms detailed in the previous chapter suffer from one major problem and that is that while they provide matchmaking between schemas they do not provide semantic matchmaking. The slight exception from this is Description Logics matchmaking which can be employed in a way that provides a certain amount of semantics, but the view of the world that it is based on is much to simple for real-world problems.

The basis for a semantic matchmaking algorithm is some form of semantic similarity measure. This similarity measure should give results that are very similar to those a human expert would give and the reasons for this should also be explainable in terms of human cognition. To achieve this it is necessary to base the similarity measure on a cognitive model.

4.1 Cognitive Grounding

Cognitive models are descriptions of how the human mind sees and organises knowledge and also how different bits of knowledge are compared to each other. From the list of existing cognitive models three will be examined more closely. The feature model introduced by [Tver77][TverGati78] and the network model introduced by [RadaEtAl89] because the semantic similarity algorithm presented in section 4.2 is based on a combination of the two. The cognitive spaces model proposed by [Gärd00][Gärd04] will also be presented due to the fact that it is a very powerful and cognitively precise model. In addition to these three models there are further models such as the alignment models of [GentMark97] and [Gold94] and hybrid models such as the one proposed by [Schw05] and [SchwRaub05] that combine aspects from different cognitive models.

4.1.1 Feature Model

The basic principle behind Tversky's feature model is that concepts are described by an unstructured list of features that together represent all the facets of the concept. Figure 4.1 gives an example of how two land-use and land-cover categories would be defined.

Based on these definitions Tversky defines a similarity measure on concepts that is the ratio between the shared features and those features that are only in one or the other concept (fig. 4.2). In the example given in figure 4.3 the common features between the Forest and the Coniferous Forest would only be the Vegetation, while for the Forest there would be one feature only used here (Trees) and the same is true for the

$$\begin{aligned}\text{Forest} &= (\text{Vegetation, Trees})\\ \text{Coniferous Forest} &= (\text{Vegetation, Coniferous Trees})\end{aligned}$$

Figure 4.1: Definitions for the categories FOREST and CONIFEROUS FOREST in the feature model.

$$sim(C_1, C_2) = \frac{|\text{Shared features}|}{|\text{Shared features}| + |\text{Features only in } C_1| + |\text{Features only in } C_2|}$$

Figure 4.2: Similarity definition for the feature model

CONIFEROUS FOREST (CONIFEROUS TREES). If these values are put into the similarity algorithm then the result is a similarity of $\frac{1}{3}$ (fig. 4.3).

This works very nicely and is very simple and easy to implement when only comparing two concepts. Unfortunately as soon as one concept is compared to two or more other concepts then problem areas in the similarity measure become apparent. The problem is that the features are completely unstructured. For example if we also compare the concept SCRUB VEGETATION (fig. 4.4) to the two concepts specified in figure 4.1 then we see that the although the similarity between FOREST and CONIFEROUS FOREST and also between FOREST and SCRUB VEGETATION is the same they actually have differences in their similarities that are not captured by the model.

An additional problem is that the features cannot be related to each other apart from via complete equality or complete inequality of their names. The fact that a CONIFEROUS TREE is a TREE cannot be modelled and thus it produces the same similarity as when compared to the SCRUB VEGETATION. Since the features are compared to each other without checking whether they are in any way interrelated this leads to strange problems when the number of unshared features is the same but they describe different attributes.

In the feature model too high an assumption is placed on the way that the features are defined. The problem above could be solved by extending the definitions as shown in figure 4.5 so that the hierarchy is represented in the feature list. This leads to a very high degree of redundancy. Specifying that a CONIFEROUS FOREST has CONIFEROUS TREES and TREES and VEGETATION adds no information to the concept and is only necessary to fix the problems inherent in the model.

Thus the feature model needs to be augmented in some way so as to solve the problems that are inherent in it.

$$sim(\text{Forest}, \text{Coniferous Forest}) = \frac{1}{1+1+1} = \frac{1}{3}$$

Figure 4.3: Similarity calculation for FOREST and CONIFEROUS FOREST in the feature model

$$\begin{aligned}\text{Scrub Vegetation} &= (\text{Vegetation, Scrub}) \\ sim(\text{Forest}, \text{Scrub Vegetation}) &= \frac{1}{1+1+1} = \frac{1}{3}\end{aligned}$$

Figure 4.4: Adding the category Scrub Vegetation to the feature model and similarity calculation

$$\begin{aligned}\text{Forest} &= (\text{Vegetation, Trees}) \\ \text{Coniferous Forest} &= (\text{Vegetation, Trees, Coniferous Trees}) \\ \text{Scrub Vegetation} &= (\text{Vegetation, Scrub}) \\ sim(\text{Forest}, \text{Coniferous Forest}) &= \frac{2}{2+1+1} = \frac{1}{2} \\ sim(\text{Forest}, \text{Scrub Vegetation}) &= \frac{1}{1+1+1} = \frac{1}{3}\end{aligned}$$

Figure 4.5: Extending the definitions with redundant information to improve the similarity calculation in the feature model.

4.1.2 Network Model

The second cognitive model that influenced the creation of the hybrid model is the network model. The different network models that exist mostly derive from semantic nets. The concepts are arranged in a graph structure as shown in figure 4.6. Some models such as [RodrEgen03] only use is-a and has-a edges in their graphs while others such as [RadaEtAl89] also allow other kinds of edges.

Similarity calculation in these models is then performed by counting the number of edges that need to be traversed to get from one of the compared concepts to the other one. This works very well when only is-a and has-a edges are considered in the model, but

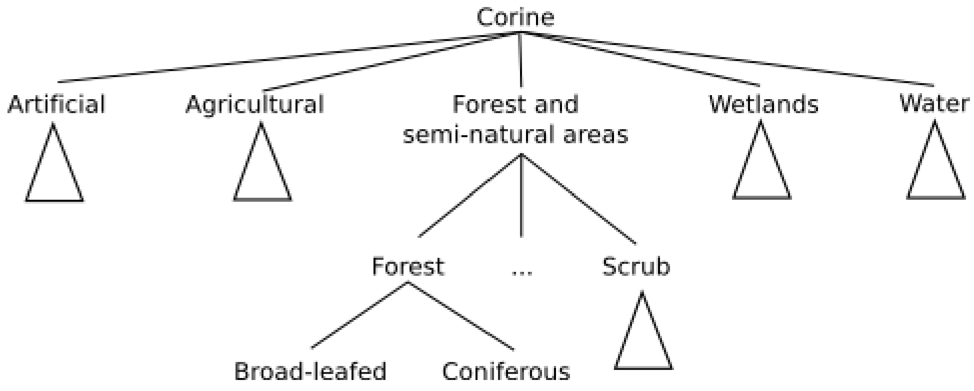


Figure 4.6: A network model of the Corine vegetation categories

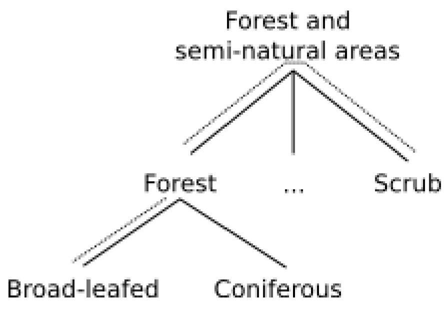


Figure 4.7: Similarity calculation in the network model

[RadaEtAl89] showed that when other edge types are used then these have to be dealt with in a way that distinguishes them from the is-a and has-a edges. If this condition is respected then the similarity measure remains cognitively sensible. When calculating distance for the concepts FORESTS, CONIFEROUS FORESTS and SCRUB VEGETATION in figure 4.7 we can see that the result is a bit better than the one derived from the feature representation of the concepts. The distance between FORESTS and CONIFEROUS FORESTS is 1 while the distance between FORESTS and SCRUB VEGETATION is 2.

Nevertheless this model is not without problems and one is that there is no internal structure for the concept nodes. So the fact that a forest consists of trees can not be modelled in this model and thus cannot be used as knowledge in the similarity calculation.

In addition to that the real showstopper when using this model for integration purposes is that it does not provide a way to integrate two graphs that are in no way connected. The two data structures to be integrated would have to be integrated by hand first and only then could the model be used to calculate the similarity between concepts in the integrated model. This defeats the basic principle of fully automatic integration. [RodrEgen03] solve this problem by combining it with a feature model to allow it to be used as an integration tool.

4.1.3 Cognitive Spaces

The third model is the cognitive spaces model proposed by [Gärd00][Gärd04]. Of the three main models this is the only one that is not in any way used in the hybrid model proposed in section 4.2, but it is included here because it is a very powerful model that can be made to very closely reflect the human cognitive model. In the cognitive spaces model concepts are points or areas in a hypercube. Each property or aspect of the concept is modelled in a separate dimension and each dimension can itself have an internal structure. The modelling of these internal structures of the different dimensions is what gives the cognitive spaces model the ability to closely reflect human cognitive models.

Similarity is then defined as either city block or Euclidean distance. City block metric is used for those dimensions that are separable and do not influence each other while the Euclidean metric is used for the inseparable dimensions. Additionally the conceptual spaces model also contains weights for the different dimensions so that the relative relevance of the different dimensions can be considered in the similarity calculation.

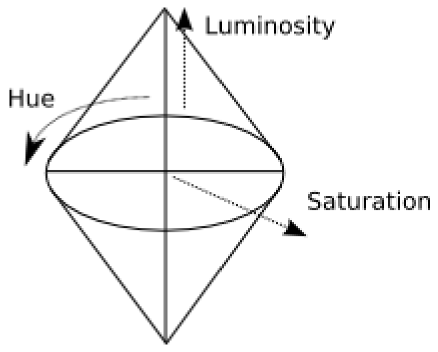


Figure 4.8: Form of the Hue, Saturation and Luminosity dimension in the cognitive spaces model.

The problems that the cognitive spaces model has are twofold. First it cannot model relations between concepts. So for an alluvial forest it cannot describe that it is a forest that lies next to a river or wetland. This could be modelled as another domain but does not fully capture the semantics of the concept. Also all dimensions apply to the complete concept. It is not possible to define that a certain dimension is only relevant for parts of the concept. That in a mixed housing/agricultural area the type of buildings is only relevant to the housing part cannot be modelled.

The second problem comes from a different direction and that is that it is hard to define the internal structure of the different dimensions. For some dimensions it is easy to define their structure such as the one given in figure 4.8 describing the human colour space or for example water speed would probably be modelled as a linear scale of meters per second flow speed.

Unfortunately for a large number of dimensions it is very hard to correctly describe these dimensions. For example what does the dimension look like that describes the different kinds of buildings or vegetation that exists. In addition it is not quite clear how to handle concepts that have no defined value for a certain dimension. Should one just ignore that dimension? What should be if the solution if the dimension has a value in the concept tom compare with? How distant is a missing dimension? These are the prime obstacles to employing the cognitive spaces model in an integration scenario. Until a clear guiding principle is given on how to model different kinds of domains and what to do with missing values it is very hard to get correct representation of the different domains and thus also very hard to get a sensible similarity value out of the model.

If these problems can be solved then this is definitely one of the most powerful models available and a prime candidate for integration purposes.

4.2 Semantic Similarity Algorithm

To create a more complete and powerful semantic similarity algorithm a hybrid model based on the feature and network models was developed. Combining these two models gives a hybrid model that is nearly as powerful as the cognitive spaces model while avoiding the problems the cognitive spaces model has with defining the internal structure of the different properties and also how to make sure that the similarities in the different properties can be compared. While it is not as complete and probably also not quite as good a representation of human cognition, these drawbacks are not too big and are outweighed by the clearer and simpler semantics and implementability.

At the uppermost level the hybrid model consists of defined concepts. These defined concepts describe the schema of the data structure. As a result of the similarity algorithm and matchmaking these defined concepts can then be tagged as most similar to another defined concept and used for the translation of data from one category to the other. The definitions of these defined concepts are based on Description Logics. These definitions are called properties and encode the semantics of the defined concepts. To be able to make any similarity calculations on the property level all properties must commit to a shared vocabulary. The shared vocabulary is stored in the skeleton ontology as a set of tree structures. Reasons for this physical structure are described in section 2.2.3.

To illustrate how the hybrid model is structured and how the similarity between defined concepts is calculated we will use schemas from the domain of land-use and land-cover catalogues. LUC catalogues consist of a collection of LUC categories that describe different types of land-use or land-cover and are usually arranged in some kind of hierarchy. The two LUC catalogues that will be used are the Austrian Realraumanalyse that was created by [Sege00] and is a very detailed classification that is geared towards the land-use categories that can be found in Austria (fig. 4.9). The second catalogue will be the European Corine[1] classification that is available for pretty much all of Europe but is much less detailed (fig. 4.10). Both figures are only excerpts from the full catalogues since displaying all categories would exceed the space available and also make the examples less clear.

4.2.1 Defined Concepts

The defined concepts form the top level of the model and describe the semantics of the schema categories that the similarity algorithm will work on, in our case LUC catalogues and categories. Each defined concept is specified using a set of necessary and sufficient conditions. Figure 4.11 shows the definitions for the Corine LUC categories MINERAL EXTRACTION SITES and INDUSTRIAL OR COMMERCIAL UNITS and figure 4.12 for the Realraumanalyse the category AGRICULTURAL AREAS.

The three LUC categories specified above showcase the basic principles of defining LUC categories. First of all every LUC category is a sub-class of the concept Area which differentiates them from the skeleton ontology concepts. Additionally every category uses the HASSOURCE property to define which LUC catalogue it comes from. Then the actual semantics are defined and in the AGRICULTURAL AREA category we see the two properties that are used most namely HASSURFACE and HASPRIMARYUSE. The INDUSTRIAL OR COMMERCIAL UNITS category then introduces the final principle that is used in the construction of definitions and that is that the ranges of properties can also be unions of multiple primitive concepts as shown with the HASBUILDING or HASPRIMARYUSE properties. These definitions then form the basis of the similarity calculation.

[1]http://terrestrial.eionet.eu.int/CLC2000

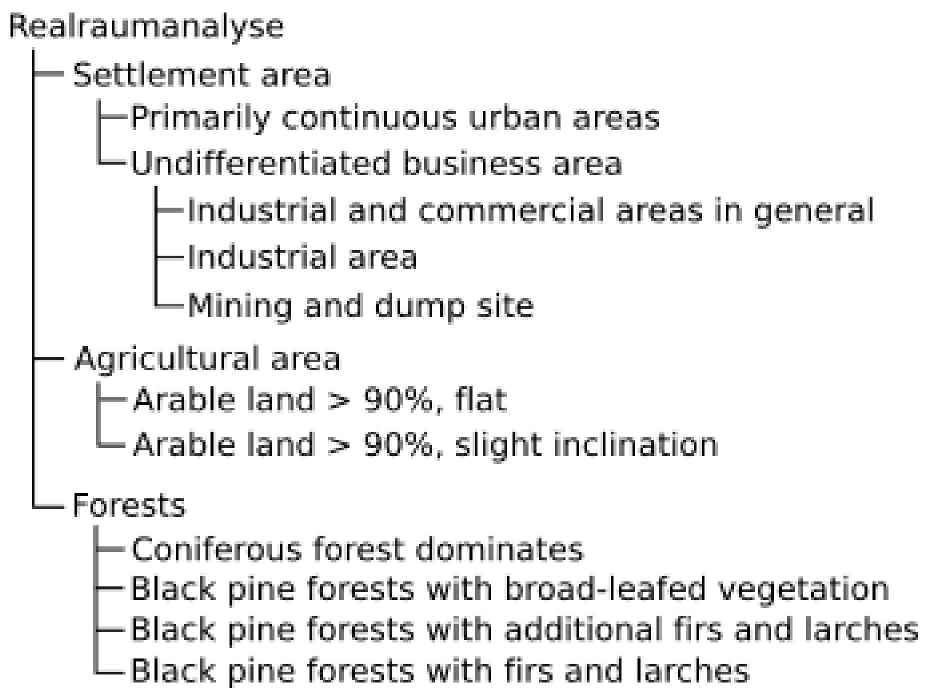

Realraumanalyse
Settlement area
Primarily continuous urban areas
Undifferentiated business area
Industrial and commercial areas in general
Industrial area
Mining and dump site
Agricultural area
Arable land > 90%, flat
Arable land > 90%, slight inclination
Forests
Coniferous forest dominates
Black pine forests with broad-leafed vegetation
Black pine forests with additional firs and larches
Black pine forests with firs and larches

Figure 4.9: Excerpt from the structure of the Realraumanalyse catalogue.

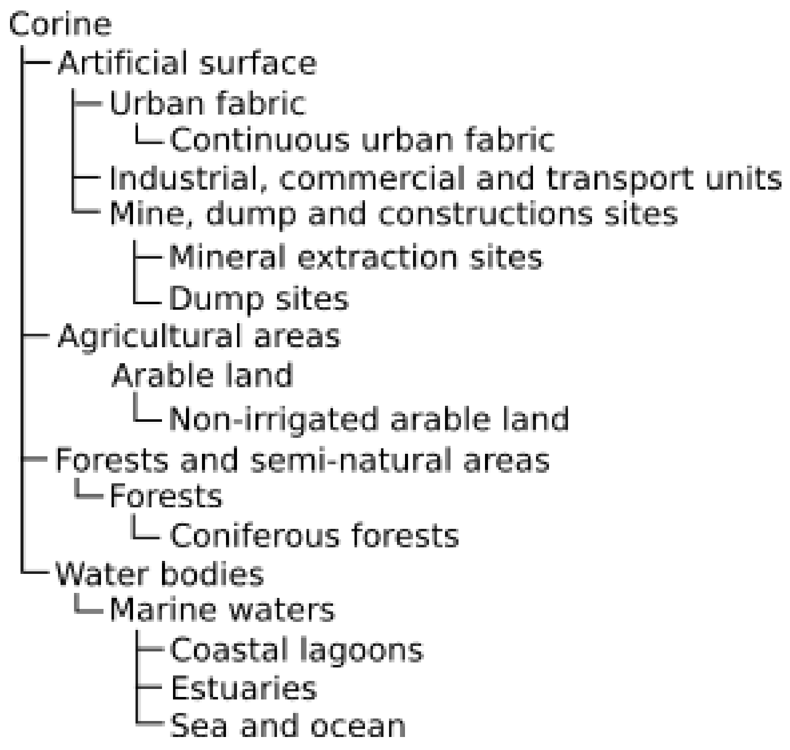


Figure 4.10: Excerpt from the structure of the Corine catalogue.

$$\begin{aligned}\text{Mineral extraction sites} &= \text{Area} \wedge \text{hasSource.Corine} \wedge \text{hasSurface.Artificial}\\ &\quad \wedge \text{hasPrimaryUse.Industrial} \wedge \text{hasBuilding.Mining}\\ \text{Industrial or commercial units} &= \text{Area} \wedge \text{hasSource.Corine} \wedge \text{hasSurface.Artificial}\\ &\quad \wedge \text{hasPrimaryUse.(Industrial} \wedge \text{Commercial)}\\ &\quad \wedge \text{hasBuilding.(Industrial} \wedge \text{Commercial)}\end{aligned}$$

Figure 4.11: Definitions for the Corine categories MINERAL EXTRACTION SITES and INDUSTRIAL OR COMMERCIAL UNITS.

$$\begin{aligned}\text{Agricultural area} &= \text{Area} \wedge \text{hasSource.Realraum} \wedge \text{hasSurface.Agricultural}\\ &\quad \wedge \text{hasPrimaryUse.Agricultural}\end{aligned}$$

Figure 4.12: Definitions for the Realraumanalyse category AGRICULTURAL AREA.

To use it to create mappings between the two ontologies the similarity measure first has to be defined for two concepts. In the hybrid model the similarity of two concepts is defined as the similarity of those properties that occur in the definitions of both concepts minus a defined value for those properties that only occur in the definition of one of the two concepts.

Figure 4.13 shows the formula for the similarity of two concepts. It results in a similarity value between 0 meaning no similarity at all and 1 meaning equality. The total similarity (S) is the similarity of the matching properties (SM) divided by the maximum possible similarity (MS). The maximum possible similarity itself is calculated by summing the weights of the matched properties and then adding a value for the unmatched properties (UP) which is the sum of the weights of the unmatched properties additionally weighted by a factor of 0.4. This formula has the desired properties that if two concepts have the same definitions then the similarity of the matched properties SM is equal to the

$$\begin{aligned}S(C_1, C_2) &= \frac{SM(C_1, C_2)}{MS(C_1, C_2)}\\ SM(C_1, C_2) &= \sum_{1}^{N} matchedPropertyWeight_n * SP(C_n^1, C_n^2)\\ MS(C_1, C_2) &= \sum_{1}^{N} matchedPropertyWeight_n + UP(C_1, C_2)\\ UP(C_1, C_2) &= (\sum_{1}^{N} unmatchedPropertyWeight_n) * 0.4\end{aligned}$$

Figure 4.13: Definition of similarity for concepts

```
function calculateSimilarity($srcOntology$, $targetOntology$)
for each $srcConcept$ in $srcOntology$ do
    $mappings = empty$
    for each $targetConcept$ in $targetOntology$ do
        $matched$ = getMatched($srcConcept$.properties, $targetConcept$.properties)
        $unmatched$ = getUnmatched($srcConcept$.properties, $targetConcept$.properties)
        $similarity$ = matchSimilarity($matched$) / maxSimilarity($matched$, $unmatched$)
        $mappings$.add($srcConcept$, $targetConcept$, $similarity$);
    end for
    $mappings$.selectBest().targetConcept.addMappedConcept($srcConcept$)
end for
```

Figure 4.14: Main loop of the semantic similarity algorithm

```
function maxSimilarity($matchedProperties$, $unmatchedProperties$)
$maxSimilarity = 0$
for each $property$ in $matchedProperties$ do
    $maxSimilarity = maxSimilarity + property$.weight
end for
for each $property$ in $unmatchedProperties$ do
    $maxSimilarity = maxSimilarity + property$.weight * 0.4
end for
return $maxSimilarity$
```

Figure 4.15: Calculation of the maximum similarity

```
function MATCHSIMILARITY($propertyList$)
similarity = 0
for each $property$ in $propertyList$ do
    $propertySimilarity = 0$
    if $property$.isConceptToConcept then
        $propertySimilarity$ = 1 - ($property$.edgeDistance / 100)
    else if $property$.isConceptToUnion then
        $propertySimilarity$ = 1 - ($property$.edgeDistance + $property$.conceptsInUnion - 1) / 100
    else if $property$.UnionToConcept then
        $propertySimilarity$ = 1 - (($property$.edgeDistance + $property$.conceptsInUnion) / 100) / 4
    else if $property$.UnionToUnion then
        if $property$.firstUnionContainedInSecondUnion then
            $propertySimilarity$ = 1 - ($property$.numberMatched / $property$.numberTotal) - ($property$.edgeDistancesSum / 100)
        else
            $propertySimilarity$ = 1 - (($property$.numberMatched / $property$.numberTotal) - ($property$.edgeDistancesSum / 100)) / 4
        end if
    end if
    if $property$.isPrimaryProperty then
        $similarity = similarity + propertySimilarity$ * $property$.weight;
    else
        $similarity = similarity + propertySimilarity$ * $property$.weight * 0.5;
    end if
end for
return $similarity$
```

Figure 4.16: Similarity calculation for the matched properties

maximum possible similarity MS of the matched concepts and the number of unmatched properties UP is zero resulting in a similarity of 1. The unmatched properties value UP is weighted lower because the missing information does not imply that the opposite is true only that nothing is specified. This reduces the similarity value of the two concepts, but the result will always show that there is at least some similarity. The similarity of matching properties SM is the weighted (WP) sum of the similarity of the ranges of the two properties (SP) which will be explained in more detail in the next chapter. The maximum similarity MS is simply the sum of all weights for the matching properties since the maximum similarity for the property ranges is always 1 and can thus be ignored and then from this the value for the unmatched properties is subtracted.

The matchmaking algorithm performs this similarity calculation for each possible pairing between concepts in the two ontologies (fig. 4.14, 4.15 and 4.16). Here it is important to note that the algorithm is asymmetric. Similarity is calculated from the source ontology to the target ontology and thus also from each source concept to the target concept. For each concept in the source ontology the similarity pairs with each concept in the target ontology are compared to each other and the pair with the highest similarity value is chosen as the final mapping. When multiple pairs with the same similarity value are found then there are two different conflict resolution strategies depending on the similarity value. If the similarity value is greater than zero then the pairing with the higher number of unmatched properties is chosen. This follows the pattern of giving higher weight to shared features than missing features. Since all pairings have the same similarity value then the pairing with the highest number of unmatched properties is the one that has the highest similarity value if the unmatched properties are ignored. On the other hand if the similarity values are all zero then the pairing with the fewest unmatched properties is chosen. The reason for this is that if a concept exists that does not have an even remotely similar concept in the target ontology then it should be mapped to the most generic concept in the target ontology to minimise the amount of error and this is the one with the fewest definitions and thus the lowest number of unmatched properties. In both cases if all pairs have the same number of unmatched properties then one pairing is chosen randomly. While this may seem to be rather crude, it is the optimum solution within the constraints of the similarity algorithm. A solution would be to add definitions to the concepts so that there are no more equivalent similarity values, but this conflicts with the premise that the minimum necessary set of definitions is used.

While it may seem that the possibility that there are two or more pairings with the same similarity value is low it does happen and one example of this is when the Corine LUC category NON-IRRIGATED ARABLE LAND is mapped into the Realraumanalyse. The Realraumanalyse has five categories NON-IRRIGATED ARABLE LAND that have additional information on the elevation of the land but no abstract category without this information (fig. 4.17) and so the Corine category is randomly mapped to one of the Realraumanalyse categories. The reason for this is the same that was given in chapter 3.2 namely that the agricultural areas in the Realraumanalyse are much more detailed and organised in a matrix, thus lacking the more abstract categories to which the Corine category would properly match. At the same time the Corine category is much more similar to the specific categories in the Realraumanalyse than to the the abstract Realraumanalyse category AGRICULTURAL AREAS.

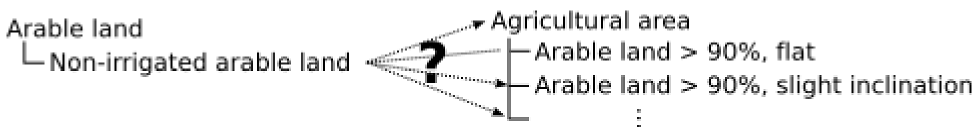

Arable land
Non-irrigated arable land
?
Agricultural area
Arable land > 90%, flat
Arable land > 90%, slight inclination

Figure 4.17: Problem of mapping the Corine Non-irrigated arable land to the Realraumanalyse.

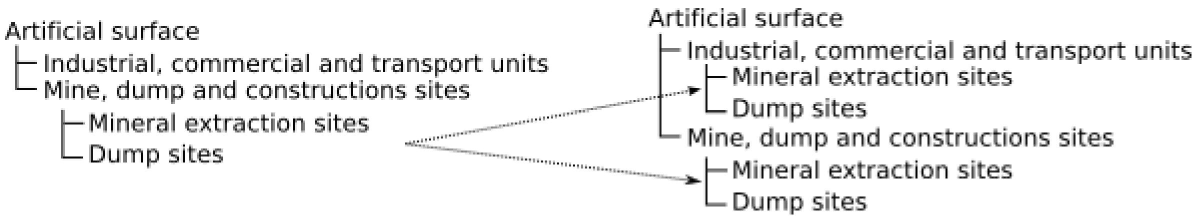

Artificial surface
Industrial, commercial and transport units
Mine, dump and constructions sites
Mineral extraction sites
Dump sites
Artificial surface
Industrial, commercial and transport units
Mineral extraction sites
Dump sites
Mine, dump and constructions sites
Mineral extraction sites
Dump sites

Figure 4.18: Reclassification of Corine categories Mineral extraction sites and Dump sites.

One other important aspect of the model is that in the ontology the defined concepts are stored as a flat list. The hierarchy is then automatically calculated from the definitions using existing automated reasoners such as RACER, FaCT or Pellet. The advantage of this is that when adding, removing or updating definitions the hierarchy is automatically updated and does not have to be maintained separately. There is also a disadvantage and that is that the calculated hierarchy is based on the semantic definitions given and this does not necessarily reflect the hierarchy that was specified by the designers of the schemas. If the calculated hierarchy and the specified hierarchy diverge then there are different reasons for this. One is that the definitions are incorrect. This is the simplest reason and can easily be fixed by correcting the definitions. The second possibility is that the original specified hierarchy is based on knowledge that has not been specified in the ontology. In this case the question arises why the knowledge has not been specified and whether this knowledge should be specified. The most problematic reason is that the originally specified hierarchy is not or not fully grounded in the semantics of the different concepts and this is both the most frequent reason and the one that cannot be solved.

Figure 4.18 shows an example of the reclassification that happens in the Corine catalogue. The categories Mineral extraction sites and Dump sites are children of the Mine, dump and construction sites category in the original classification, but since they are both industrial complexes and are also defined as such the reasoner places the concepts under the Industrial, commercial and transport units category as well. This is an example of making implicit knowledge explicitly visible.

Figure 4.19 shows a similar example from the Realraumanalyse which demonstrates the case where the original hierarchy does not really take the semantics of categories into account. The original flat list is transformed into a four level hierarchy that more closely follows the semantics of the categories.

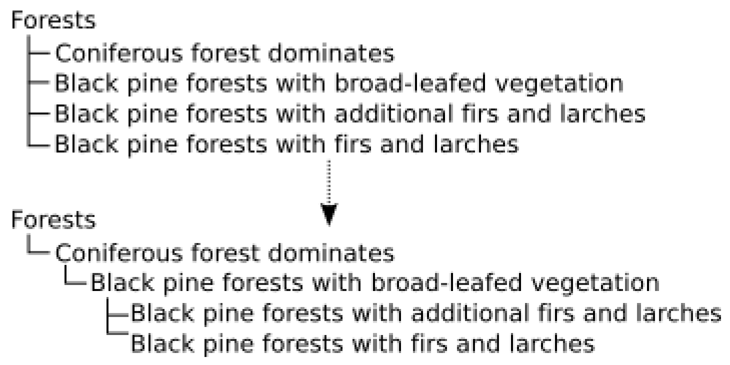

Forests
Coniferous forest dominates
Black pine forests with broad-leafed vegetation
Black pine forests with additional firs and larches
Black pine forests with firs and larches
Forests
Coniferous forest dominates
Black pine forests with broad-leafed vegetation
Black pine forests with additional firs and larches
Black pine forests with firs and larches

Figure 4.19: Reclassification of the FOREST tree in the Realraumanalyse.

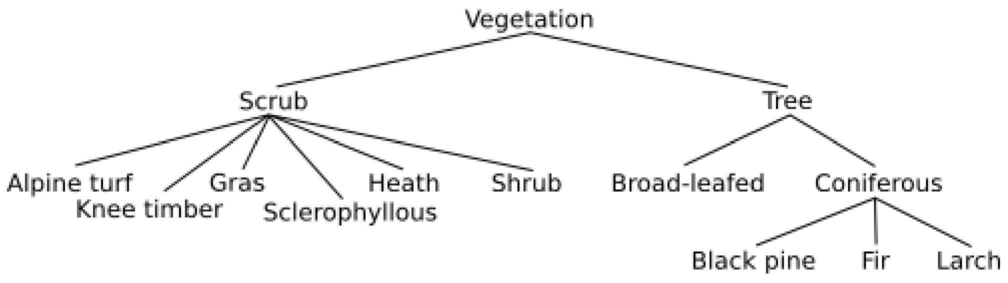


Figure 4.20: Part of the skeleton ontology describing vegetation.

4.2.2 Skeleton Ontology

In order to make it possible for two defined concepts to be compared, they must commit to a certain shared vocabulary. This shared vocabulary defines the properties that can be used to define the defined concepts and what values can be used in the definitions. Since multiple ontologies can be defined based on the shared vocabulary it is impractical to include it in every ontology. For this reason it has been factored out into the so called skeleton ontology.

It is organised in a set of trees with every tree representing the hierarchical structure of a certain area of knowledge. Figure 4.20 shows the tree structure describing the knowledge about vegetation. At the top it splits into two main areas scrub and tree vegetation. The scrub sub-tree contains all vegetation types that do not fulfil the criteria for being trees. The second area contains the information about those tree types that are necessary to describe the LUC categories that are defined.

The principle that only the minimum information that is necessary to describe the defined concepts is encoded is a guiding principle that is followed throughout the modelling of the domain knowledge. For example while there are plenty of broad-leafed trees and some of them also grow in the forests of the described areas, this information is not added to the skeleton ontology, because there are no LUC categories that would make use of this information. The information about broad-leafed trees could be added to the skeleton ontology, but since it would provide no additional information and only increase the complexity of the skeleton ontology it is left out. The same goes for the definitions of the LUC categories. For example such information as that an area of arable land has some kind of access road, because otherwise the tractor could not reach it and thus it would not be arable or that it needs to be harvested at some time is not included, because again while it is information about the LUC category it is not information that defines the LUC category. On the other hand the fact that an area of arable land in the Corine catalogue should be homogeneous is included because in the metadata about the LUC category it is explicitly stated that to be classified as arable land it must be homogeneous otherwise it is classified differently.

For each tree describing a certain area of knowledge at least one property is defined. The range of the properties is set to the root node of the area of knowledge that they belong to. This of course implies that any children of the root node are also valid values for the property. While in most cases one property is sufficient for example in the area of vegetation it is useful to differentiate between the primary and secondary vegetation. Thus two properties are defined that have the same range but are ordered and thus also differently weighted in the similarity algorithm.

As described in the section 4.2.1 when the definitions of two defined concepts are compared the similarity of two matching definitions is the similarity of their ranges. Since the range definition allows for the union of concepts the similarity algorithm for concepts must support four different situations

- Both ranges are single primitive concepts.
- The source range is a primitive concept and the target range a union of primitive concepts.
- The source range is a union of primitive concepts and the target range a single primitive concept.
- Both ranges are unions of primitive concepts.

For each of these situations we will now look at how the similarity is calculated.

Comparing two primitive concepts

Comparing two primitive concepts is the easiest of the comparison cases. The similarity of two primitive concepts is inversely proportional to the minimum number of edges that must be traversed in the skeleton ontology to get from the one concept to the other (fig. 4.21). Figure 4.22 shows the similarity calculation for the pair Broad-leafed and Coniferous trees. To get from Broad-leafed to Coniferous requires traversing two edges so the similarity SP between the two categories is 0.98. What is important for this to return valid results is that there is only one branching principle for each level in the tree of primitive concepts. This has the effect that siblings are of the same quality and that each edge between a sibling and the parent concept has the same similarity distance. If this were not true then the similarity values of two siblings with their parent could be the same even though they actually have differing similarities.

Comparing a primitive concept to a union of primitive concepts

Comparing a primitive concept to a union of primitive concepts is similar to comparing two primitive concepts. The primitive source concept is compared to each of the primitive

$$SP(C_1, C_2) = 1 - \frac{|\text{edges traversed}|}{100}$$

Figure 4.21: Definition of similarity for two primitive concepts.

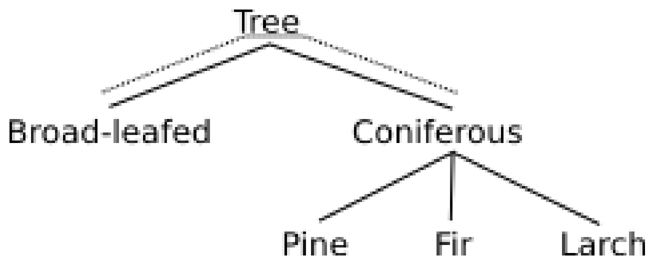


Figure 4.22: Similarity calculation for the two primitive concepts BROAD-LEAFED and CONIFEROUS.

concepts in the union using the primitive concept to primitive concept similarity calculation. From these comparisons the best one is chosen and the result from that comparison is weighted to compensate for the additional primitive concepts in the union and that is the final similarity result (fig. 4.23). Figure 4.24 gives an example of the comparison of the MINING building to the union of INDUSTRIAL and COMMERCIAL buildings. The nearest of the two is one edge away thus the similarity SP would be 0.98.

Comparing a union of primitive concepts to a primitive concept

This comparison is calculated by switching the two parameters and then applying the primitive concept to union of primitive concepts calculation (fig. 4.25). The result is then weighted by a factor of 0.25 that signifies the fact that a lot of information is lost in this mapping. The factor has been determined empirically and further study is necessary to guarantee that it is valid in other domains as well.

Comparing two unions of primitive concepts

The final comparison is the one that delivers the widest range of similarity values. While the first two comparisons tend to result in high similarity values and the third one rather lower ones this comparison gives results that range from complete matches to very low similarity. It is calculated in the following way. First the optimal mapping between the primitive concepts of both unions is calculated using the primitive concept to primitive concept calculation. Then the rough similarity is calculated as the ratio between the

$$SP(C_1, U_2) = 1 - \frac{|\text{edges to nearest}| + |\text{unmatched concepts}|}{100}$$

Figure 4.23: Similarity calculation for a primitive concept to a union of primitive concepts.

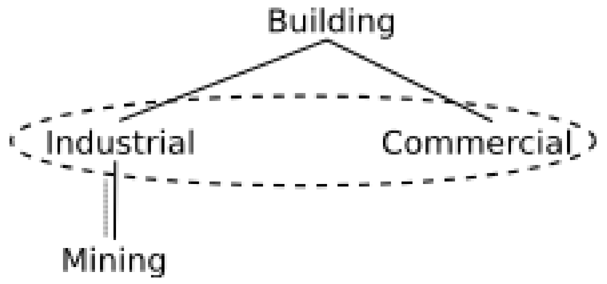


Figure 4.24: Similarity calculation for the concept MINING to the union of INDUSTRIAL and COMMERCIAL.

$$SP(U_1, C_2) = \frac{SP(C_2, U_1)}{4}$$

Figure 4.25: Similarity calculation for a union of primitive concepts to a primitive concept.

number of matching and non-matching concepts. Finally this rough similarity is modified by the similarity results from the matching concept to concept comparisons (fig. 4.26). The example in figure 4.27 show the similarity between the union Service building and camp site to the union of Service and Recreational building. All concepts can be matched thus the similarity is 1, but not all matches are perfect and the inexact factor is 0.01 in this case making the total similarity SP 0.99.

4.3 Evaluation

Many similarity algorithms are shown to work with certain small data sets, but to truly test a similarity algorithm it is necessary to apply it to a larger real-world data set. This has been done for the similarity algorithm presented in the previous chapter. It has been applied to the domain of land-use and land-cover catalogues in the region of Carinthia, Friuli-Venezia Giulia and Slovenia. These three regions have more or less strongly differing LUC catalogues. The Carinthian Realraumanalyse has been designed from the ground up to describe the Austrian land-use situations and is very detailed. At the other end the Slovenian data is in the European Corine catalogue that is designed to be used only

$$\begin{aligned} SP(U_1, U_2) &= \frac{|\text{matching concepts}|}{|\text{total concepts}|} - \text{inexact factor} \\ &\quad \text{(if the first union is contained in the second one)} \\ SP(U_1, U_2) &= \frac{|\text{matching concepts}|}{|\text{total concepts}| * 4} - \text{inexact factor} \\ &\quad \text{(if the second union is contained in the first one)} \end{aligned}$$

Figure 4.26: Similarity calculation for two unions of primitive concepts.

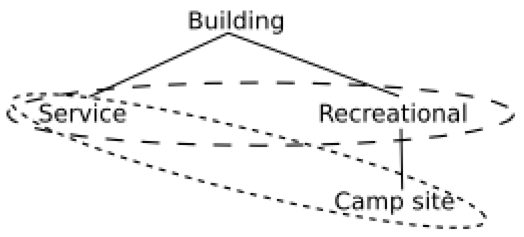


Figure 4.27: Similarity calculation for the union of SERVICE and CAMP SITE to the union of SERVICE and RECREATIONAL

with satellite photos and is thus very coarse. In between lies the Italian Moland catalogue that is based on Corine but is more detailed in the size of the polygons in the data set and has also been augmented with further categories that are of interest to the Friulan users. When applying the model and similarity algorithm to the three LUC catalogues the following conclusions can be made

- the algorithm works sufficiently well to be used in real-world situations,
- performance is a bottleneck,
- the resulting mappings are asymmetric,
- the algorithm rather maps to more general than to more specific categories,
- some aspects cannot be modelled properly within the constraints of the current system,
- in some areas human input might be sensible.

These conclusions will now be analysed in detail.

4.3.1 Tested in Real-World Situations

The results of applying the semantic similarity algorithm to the LUC catalogues have been evaluated by experts from the Institute for Geography and Regional Studies at the University of Klagenfurt[2], the Institute for Geoinformatics at the University of Münster[3] and the Spatial Data Infrastructures Unit at the Joint Research Centre[4].

The evaluation results show that out of a total of 200 mappings between the Realraum-analyse and Corine 197 mappings are correct with only three erroneous mappings caused by shortcomings in the model and similarity algorithm. Of these 197 correct mappings 30 were found that while correct would have been handled differently by the experts due to different methodologies. In addition to these the evaluation also showed up a number of modelling errors ranging from incorrect and missing knowledge to errors resulting from incorrect original metadata.

The results of the evaluation of the mappings between the other ontologies in the system revealed a similar pattern with the average error rate varying between 0 and 5%. This

[2]http://www.uni-klu.ac.at/geo
[3]http://ifgi.uni-muenster.de
[4]http://ies.jrc.cec.eu.int/sdi.html

clearly shows that both the cognitive model and the semantic similarity algorithm are sufficiently capable and correct to be aplied in real-world situations. Nevertheless the rate of modelling errors of about 8% makes it clear that an implementation in a fully unsupervised situation is not possible since the number of modelling errors is too high and these errors can only be found by reviews and evaluations.

Shortcomings of the model

Unfortunately there are some erroneus mappings due to shortcomings of the model. These are the most serious errors since the model needs to be changed and extended to solve them. One example for this category is the Realraumanalyse category Other non-built-up areas belonging to the public administration which in order to be fully and correctly defined would require the use of a negation operator since it must define that it does not contain buildings and this operator currently does not exist in the model. Another example is the Realraumanalyse category Knee timber partially with turf or rocks. Here the model does not provide the primitives to define the relations between different properties of the category. It is not possible to define that the surface is a combination of natural vegetation and rocks with the ratio being around 90 to 10 percent. The same would have to be defined for the ratio between knee timber vegetation and alpine turf. While it is possible to define a primary and a secondary vegetation or surface it is impossible to define what ratio exists between the two vegetation types. Since the semantics cannot be correctly modelled the similarity algorithm then miscalculates and in this case assigns the category to the Corine category Bare rocks.

Adding further primitives to the model is hard since it then also requires defining how these primitives influence the similarity calculation. Still it is necessary so that the similarity algorithm can be applied to more complex situations. Further shortcomings of the model that were already known before the evaluation are discussed in section 4.3.5.

Correct mappings that would be handled differently by the domain expert

When the algorithm was developed one of the core principles was that it should provide a complete mapping from one ontology onto another. If there is no really similar concept in the target ontology then the algorithm should find a similar but more abstract concept to which to create the mapping to. The extreme being the situation where a concept is mapped to the root concept of the target ontology. Examples of such mappings are from the Corine categories Sea and Ocean, Olive groves or Annual crops associated with permanent crops. For these categories there are no similar categories in the Realraumanalyse since these things do not exist in Austria and the algorithm correctly maps them to higher level categories. The problem is that the domain experts are able to detect these situations where there is no truly similar category and then apply a different strategy than the strategy used by the system. One of the strategies suggested by the expert was to simply not create mappings for those categories for which there are no proper similar categories and so create blank areas in the map. Anther strategy was to change the mapping to a different category based upon knowledge that was not contained in the original metadata but which came from experience dealing with land-use catalogues

Ontology	# Concepts	Avg # properties	Load time
Corine	64	3	31sec
Moland	96	5	3min 19sec
Realraumanalyse	136	6	5min 16sec

Table 4.1: Load times for the ontologies in the system

and data sets. While in some cases this knowledge could be added to the definitions in most cases it came down to a personal hunch and that cannot be modelled.

4.3.2 Analysis of Algorithm

It is very likely that the semantic similarity algorithm is applied to ontologies of different sizes. Especially when dealing with large ontologies this means that the complexity of the algorithm becomes a relevant question. Basically the algorithm matches all concepts from the source ontology to the target ontology and then selects the best mapping. This means that when comparing two ontologies the maximum number of comparisons that have to be made is the number of concepts in the source ontology times the number of concepts in the target ontology times the number of properties that exist ($\mathrm{O}(N \cdot M \cdot P) = \mathrm{O}(N^3)$). So the complexity of comparing two ontologies is cubic. In addition to this each ontology has to be compared to all the other ontologies in the system and since the algorithm is asymmetric for each ontology pair two calculations need to be made ($\mathrm{O}(\frac{N \cdot (N-1)}{2} \cdot 2) = \mathrm{O}(N^2)$). This results in a total complexity of $\mathrm{O}(N^5)$. While this is already pretty slow it is further increased by the fact that the ontologies are loaded and the hierarchy inferred using Description Logics. The Description Logics inference algorithm runs in exponential time thus making the whole system run in exponential time.

Running times

Table 4.1 shows the three ontologies that are currently available in the system, the number of concepts in each ontology, the average number of properties per concept and the time the Description Logics algorithm takes to load them. Table 4.2 shows the times that the comparisons between two ontologies take. These numbers have been calculated by running the algorithm which is implemented in Java[5] five times on a PowerPC G4 1.5GHz with 768MB RAM and then averaging the run times. Since in a production environment the ontologies would not be mapped to themselves the complete similarity calculation takes about 2 minutes and the loading operation about 9 minutes. Thus the majority of the total running time of 11 minutes is spent in the Description Logics reasoner.

The timings in table 4.1 clearly demonstrate the exponential increases in the time required to infer the hierarchy. Increasing the number of concepts by a factor of 1.5 as when moving from the Corine to the Moland ontology leads to an increase of the running time by a factor of nearly 8. This increase is mainly due to the fact that when more concepts are in an ontology then each concept must be defined using more properties and this number

[5]http://java.sun.com

From / To	**Corine**	**Moland**	**Realraumanalyse**
Corine	5sec	10sec	15sec
Moland	11sec	20sec	31sec
Realraumanalyse	18sec	34sec	52sec

Table 4.2: Running times for the comparison of two ontologies

of properties is the primary influence on the running time. Between the Moland and Realraumanalyse ontologies the average number of definitions does not increase so much and thus the running time only increases by a factor of about 1.5.

An interesting fact that is visible in table 4.2 is that comparing a more complex ontology to a less complex one is slower than in the opposite direction. Again the reason for this is the average number of properties per concept since if the number of properties is lower in the source ontology then the calculation of missing properties is slightly faster. Also clearly visible in table 4.2 is that roughly doubling the number of concepts and properties per concept leads to an increase of the comparison time by a factor of 10.

The timings shown in the two tables clearly indicate that it is not possible to run the algorithm in an interactive way since the response time would be much to high. While this is problematic as it means that the algorithm cannot be easily implemented as an interactive web service that takes two data sources and the ontologies that describe them and produces a mapping between them since the run time of possibly hours or days would exceed the time-out period for an interactive service. It is nevertheless possible to use it in an half interactive situation where the mappings between the ontologies are precalculated offline and then the service only needs to take the actual data and can translate the data from one classification into the other. This is the approach that has been taken in the current implementation of the system. Still the fact that the Description Logics part runs in exponential time means that the system cannot be used for larger ontologies since then the system running time reaches the point where the algorithm for all practical uses never terminates.

Optimisation

Clearly the complexity and speed of the system need to be improved and the area that offers the best possibility of improvement is the process of loading the ontology hierarchies. The best solution to achieve an improvement in this area would be to remove the concept hierarchy inference and statically define the concept hierarchy when modelling the ontologies. While this also removes the benefits provided by the hierarchy inference such as easier maintenance and also a first check of the correctness of the definitions the speed improvement provided by eliminating this step simply outweighs the negative aspects. With this optimisation the complexity of the complete system would immediately be reduced from exponential time to polynomial time which while still very high means that at least the algorithm will run in a time that can be handled in real-world situations

Optimising the similarity calculation itself is much harder. One possibility would be to use some kind of heuristic to determine which concepts to compare or at least which concepts not to compare. Section 3.2 showed that at least for land-use data sets instance

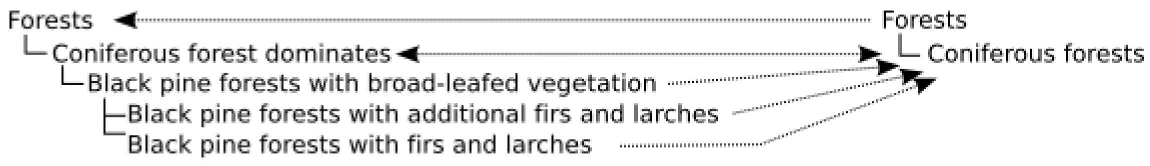

Forests
Coniferous forest dominates
Black pine forests with broad-leafed vegetation
Black pine forests with additional firs and larches
Black pine forests with firs and larches
Forests
Coniferous forests

Figure 4.28: Example of asymmetry when mapping from the more detailed Realraumanalyse to Corine

statistics cannot be used, but they might work in other domains. The heuristic would of course have to be admissible so that the optimal mapping is always found.

The algorithm could also only expand the best similarity match at any level. In this case the algorithm would first calculate the similarity values for all concepts at the top level of the hierarchy and then only expand into the children of the best match. This process would then be repeated until the algorithm reaches the leaves of the hierarchy. While this would increase the speed of the algorithm it also relies on the fact that the hierarchy is based upon the semantic definitions and thus is liable to make incorrect choices if the hierarchy is statically defined by the ontology creator. The possibility of a wrong choice can be reduced by not only expanding the best child but the best N children, but even then this optimisation would mean that the algorithm is no longer guaranteed to find the most similar concept.

One optimisation that the algorithm definitely lends itself to is parallelisation. The similarity calculations and values of one concept do not influence those of another concept. Thus it is easily possible to perform them in parallel. This does not improve the actual complexity of the algorithm but in practise could reduce the running time of the system.

4.3.3 Asymmetry of resulting mappings

An important aspect is that the results from the similarity algorithm are asymmetric. This means that if a mapping is traversed from a concept in the Corine catalogue to a concept in the Realraumanalyse category and then back then it is not guaranteed that the Corine category is the one that was started from. This asymmetry has two sources. One is that the algorithm itself is asymmetric when comparing property ranges. This means that often comparing two concepts gives different values depending on which category is used as the source category. This can mean that in the case where the similarity value is lower there can be a different mapping with a higher similarity value. The second reason for the asymmetry lies with the data. It is often the case that one of the two LUC catalogues that are compared is more ore less detailed in that area. A nice example of this is the detail level of forests in the Realraumanalyse as shown in figure 4.28. Here the more detailed categories of the Realraumanalyse are mapped to the more general category in Corine which in turn is mapped to the more general category in the Realraumanalyse. Figure 4.29 showcases the second reason why asymmetry can arise from the data, because the Realraumanalyse does not have any categories related to seas and thus those categories are mapped into the Realraumanalyse category OTHER AREAS.

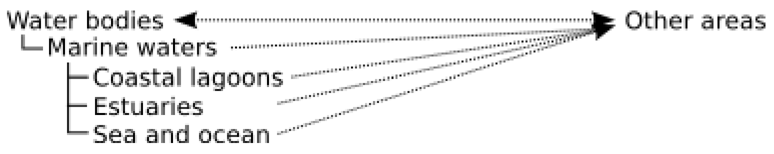


Figure 4.29: Example of asymmetry due to missing categories when mapping from Corine to the Realraumanalyse.

4.3.4 Mapping to more general categories

One interesting aspect that has become apparent from the use of the system is that the similarity algorithm has a tendency to map categories to categories of a similar or higher level of abstraction rather than to one of lower abstraction. This is actually the most ideal characteristic that a similarity algorithm can have. What it means is that when traversing the mappings there will probably be some information loss but no information will be magically added. This is to be preferred over the situation where a more abstract concept is mapped to a more concrete concept. In this situation if the user only sees the mapped data and has no way of seeing the mapping then it is possible that the user draws conclusions on the translated data that are based on the information that is added by the mapping and not actually present in the original data. The prime example is the one touched upon in previous sections where the Corine category Non-irrigated arable land is mapped onto the Realraumanalyse category Non-irrigated arable land on a flat surface. The information about the slope such as flat in this case is not a actually available in the Corine data, but when viewed through the mapping to the Realraumanalyse it seems to be there and wrong conclusions could be based on it. Currently no solution for this problem exists and it might be one of the areas where human input might be necessary.

4.3.5 Shortcomings of the model and similarity algorithm

In addition to the shortcomings that were found a part of the evaluation process there area a few shortcomings that were already known but that do not influence the similarity calculation. One such shortcoming can be seen in the Realraumanalyse category Alluvial Forest. An alluvial forest is a forest that lies next to a river or wetland. Since the model currently does not support relations between concepts the fact that an alluvial forest is a category that must lie next to a river cannot be fully modelled. Currently this problem is fixed with a workaround that adds a property lies next to to the skeleton ontology with ranges such as River. But actually this does not fully capture the semantics since the the alluvial forest does not lie next to any river but only next to rivers as they are defined in the Realraumanalyse. These kinds of spatial relations between categories probably need to be added to the model and the similarity algorithm extended in such a way as to support such constructs.

In addition to the fact that relations between schema concepts cannot be modelled it is also impossible to define relations between concepts in the skeleton ontology. One area where this would be interesting is the different types of slope gradient that are defined in the skeleton ontology. These gradients are currently arranged as siblings in the gradient

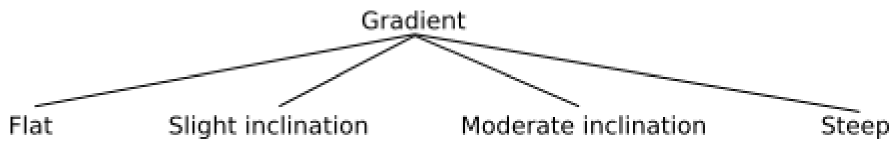


Figure 4.30: Current model of gradients in the skeleton ontology.

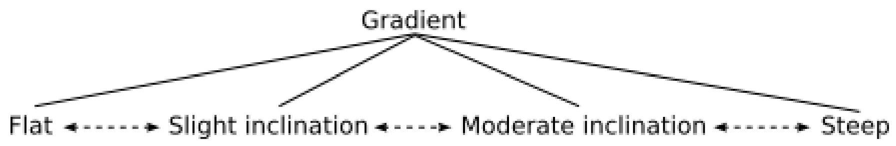


Figure 4.31: Extended gradients model including direct relations between siblings.

tree (fig.4.30). While this is sufficient for the current situation since the gradient is only used in the Realraumanalyse and thus never influences the similarity calculation except as a missing property if it is used in other catalogues as well then this creates a problem. What happens is that since the gradients are all siblings the similarity between each pair of gradients is exactly the same. This is obviously not true since a flat slope is much more similar to a slope with a slight inclination than to a steep slope. Adding a relation that orders the gradients (fig. 4.31) to the model leads to the question how this changes the similarity algorithm. What should the algorithm do if it encounters such a relation. Does it mean that the is-a hierarchy cannot be used at all at this level in the tree or is the is-a relation weighted so that it would lead to a stronger difference than the relation between siblings. Also is a traversal of one edge of this new relation the same as traversing one is-a edge when adding the similarity results from a property with such relations to one that uses only is-a relations. Further research is necessary to answer these questions and thereby improve the model and similarity algorithm.

4.3.6 Human input

Finally the question remains whether in some areas human input into the similarity calculation might be a sensible solution. Originally the idea was that the similarity algorithm should be fully automatic and require and allow no human input apart from the original modelling work. While this seemed like a good idea at the outset of the development of the algorithm the experiences from using the algorithm and also from the evaluation suggest that the capability for additional human input would be useful. It would make it possible to correct those mappings that the expert evaluation has determined to be wrong, but could also used to change those mappings that the domain experts thought should be changed even though they were correct. Since the evaluation showed such a large number of errors stemming from the modelling process it is clear that a fully automatic implementation is unrealistic and as such the human input could easily be integrated as part of the evaluation process.

5 HarmonISA Land-use Data Integration System

One of the core tenets of the European union is the concept of a Europe of regions that span across territorial lines. Apart from all other efforts this requires that the data and data structures of the co-operating regions are integrated so that there are no longer dividing boundaries at the national borders. The EU funds projects aiming to improve this co-operation and one such project is ISA-Map[1]. ISA-Map aims to harmonise the national and regional data resources to enable transnational planning between Carinthia, the autonomous region Friuli-Venezia Giulia and Slovenia. Different sub-projects exist within the ISA-Map project ranging from harmonisation of metadata, a viewer for different geodata covering the region and the HarmonISA[2] project which deals with semantic integration of land-use data.

The HarmonISA project which was carried out at the Institute for Geography and Regional studies at the University of Klagenfurt forms the framework within which the semantic similarity algorithm presented in section 4.2 was created. The aim was to develop a set of tools to automatically integrate different land-use data sets to create a homogeneous view onto the land-use of the whole region.

5.1 Inside the HarmonISA System

The HarmonISA system provides an integration environment for LUC data. It does this using a combination of existing tools and libraries combined with an implementation of the semantic similarity algorithm and the results of which are then applied to the actual data.

5.1.1 Technical Information

All tools and libraries that have been developed in the HarmonISA project have been implemented in the Java[3] programming language. The reason for this is that Java is a cross-platform development tool and since the HarmonISA system is designed to run on

[1]Harmonisation of regional data resources for cross-border planning, a project within the INTERREG IIIB CADSES framework, http://www.isamap.info
[2]http://harmonisa.uni-klu.ac.at
[3]http://java.sun.com

the systems of three different countries no assumptions can be made about the platforms on which the tools will have to run. Another advantage of the Java language is that it provides a wide array of libraries that provide required functionality. These libraries made it possible to quickly develop the core functionality of semantic similarity reasoning without having to worry about such functions as loading OWL ontologies[4] [BechEtAl03], Description Logics reasoning[5] or handling geographical data[6].

The HarmonISA system consists of one big web application that provides the main interface into the land-use data and similarity information and a set of small tools that support the management of the system. Before we look into the management of the system a short excursion into the area of LUC data.

5.1.2 Land-use / Land-cover Data Sets

This thesis has previously dealt with the problems that arise when attempting to formalise the semantics of land-use and how to deal with the differences between the different land-use catalogues. When transitioning from dealing with land-use data on the schema level to land-use data on the instance data additional problematic areas arise.

Land-use / land-cover data sets are acquired by digitising and interpreting satellite and aerial photographs. Depending on the scale at which the photos were taken the resulting data sets have a different granularity. An aerial photograph at a scale of 1:5000 will allow the borders of a forest to be marked out much more precisely than if based on a satellite photo with a scale of 1:100000. Also digitising the borders of any area of land-use is a process of approximating the true shape of the area by a polygon. The more detailed the photograph the more detailed the polygon can be and the better the actual land-use pattern can be approximated. Finally each vertex of the polygon has a certain geographical position on the earth. These coordinates are assumed to be on the spherical earth but if they are to be displayed on a flat two-dimensional map the coordinates need to be projected. Different types of projection exist and different countries use different projections due to historical but also precision reasons. When integrating data from multiple sources these projections need to be harmonised as well and this adds another source of errors to the geographical data. The results of these small errors can be seen on the border between Carinthia and Slovenia (but of course also on the other borders) as white areas where there is no land-use at all (fig. 5.1). These situations are inevitable and can only be solved by correcting each of the affected polygons by hand.

One further problematic situation exists and that is that in some cases the different LUC catalogues have different cut-off values for when a certain land-use is included in the data set and for when the area is just added to the surrounding land-use category. An example of this is in the Corine catalogue where URBAN FABRIC is only considered if the total area of that part is over 4 hectare. When data from a more detailed catalogue such as the Realraumanalyse is translated into the Corine catalogue these cut-off and amalgamation rules are not considered and thus the result is classified as Corine data but the level of

[4]OWL API: http://owl.man.ac.uk/api.shtml
[5]Pellet DL reasoner: http://www.mindswap.org/2003/pellet/index.shtml
[6]GeoTools geoprocessing library: http://docs.codehaus.org/display/GEOTOOLS/Home

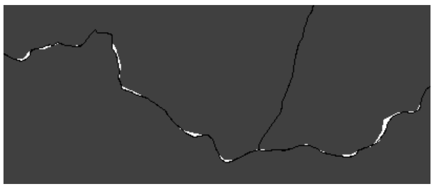


Figure 5.1: Errors in the border area between Carinthia and Slovenia

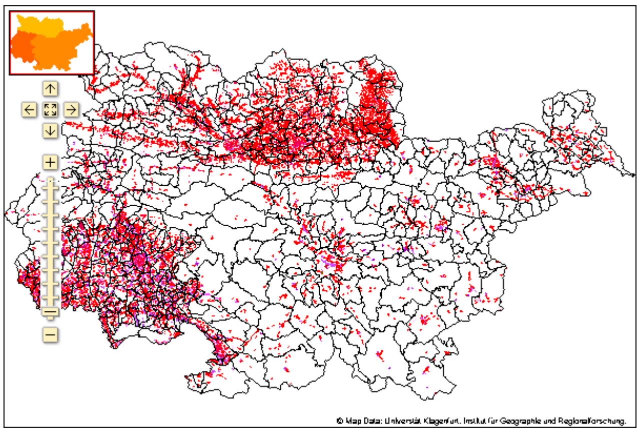

© Map Data: Universität Klagenfurt, Institut für Geographie und Regionalforschung.

Figure 5.2: Urban fabric in the three regions

detail is much higher. While this may not seem like a problem at first it is troublesome when comparing this much more detailed data to original Corine data. Figure 5.2 shows Urban Fabric in the three regions and it is clearly visible that in Carinthia and Friuli-Venezia Giulia the detail level at which urban fabric is acquired is much higher than in Slovenia. If a user is not aware of the fact that there are these differences in the original data then he could conclude from the map that Slovenia is basically not very populated at all. What the user cannot see is that if Corine data for Carinthia would be used it would look similar to the data from Slovenia.

5.2 Creating the Integration Mapping

A lot of steps are required to get from a set of unconnected data sets to the fully integrated view the user gets through the HarmonISA application (fig. 5.3). The data sets and their metadata need to be analysed, their semantics extracted, these semantics then need to be encoded in ontologies and if necessary the skeleton ontology has to be expanded to provide the tools to describe the semantics, then the ontologies are fed into the similarity algorithm, the results are evaluated and if errors are found then the process starts from the beginning again. While it is possible to enter the complete data sets into the ontologies in one iteration it is preferable to only encode a small portion of the LUC categories at first and then iteratively slowly add the remaining categories. The advantage of this is that it breaks down the task into manageable parts and reduces the number of errors that are made, since it allows to focus on one part at a time and do that one properly.

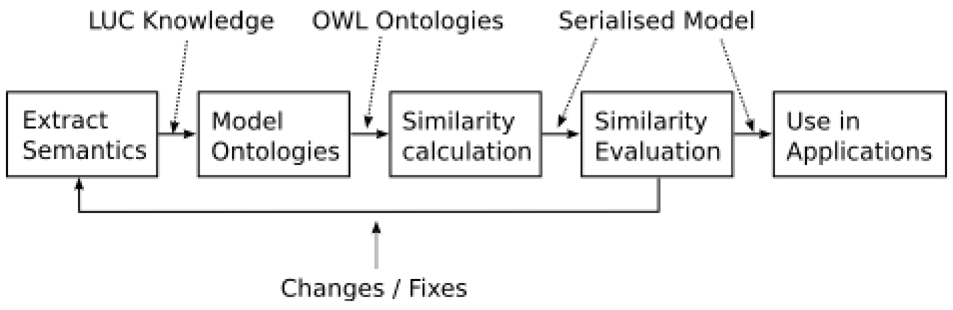


Figure 5.3: Integration mapping workflow

5.2.1 Extraction of Land-use Category Semantics

The first step is analysing the data sets and metadata to extract the semantics of each LUC category. Depending on the data source more or less metadata is available. The Corine catalogue has a very extensive metadata documentation that very clearly defines which land-use the categories describe. At the other end of the spectrum is the Realraumanalyse which basically only has the category names as descriptions. The advantage with the Realraumanalyse was that it was also developed at the Institute for Geography and Regional Studies in Klagenfurt and thus it was possible to ask the creators when the semantics of a category were unclear. It is important that this step is not rushed since everything from here on depends on the fact that the semantics of the LUC categories have been extracted correctly.

5.2.2 Modelling Semantics in OWL Ontologies

The first step and the second step of encoding the semantics in the ontologies are not as clearly delimited as described here and happen pretty much in parallel. As mentioned earlier the chosen ontology language was OWL and since OWL files are very verbose XML files it is recommended to use some tool to create the ontologies. The primary tool for creating OWL ontologies is Protégé[7] developed at the University of Stanford. While Protégé was primarily developed for its own ontology language it provides a very powerful plug-in architecture and together with the OWL plug-in[8] can be transformed into a really powerful editor for the OWL language (fig. 5.4). The advantage of using this tool is that it provides a set of functionalities that are very useful in the ontology modelling process and help to avoid making errors.

- It sports an *inline editor for the restrictions* that define the LUC categories (fig. 5.5). The advantage of using this editor is that it catches all syntactic errors while the definitions are being created. Thus it is impossible to create definitions that are syntactically wrong which is an improvement since it means that these errors are caught earlier on and not only when trying to calculate the mappings.
- It provides simple *wizard functionality* that aids in the development of the skeleton ontology. The wizards provide such functionality as creating multiple subclasses

[7]http://protege.stanford.edu
[8]http://protege.stanford.edu/plugins/owl/

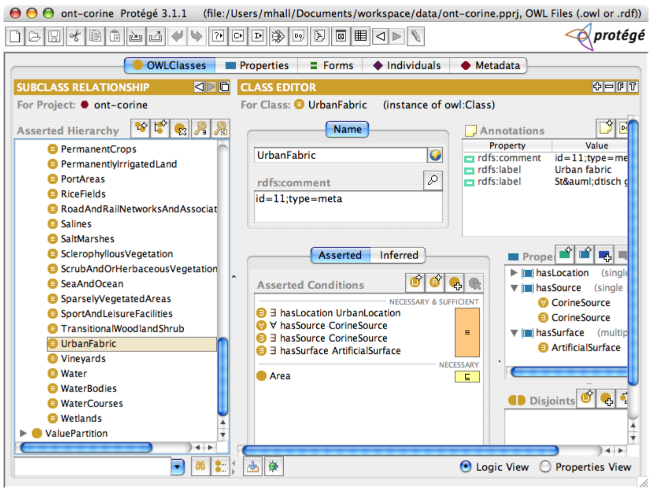


Figure 5.4: Protégé Ontology Development Environment

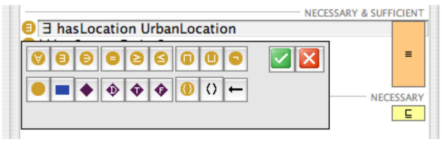
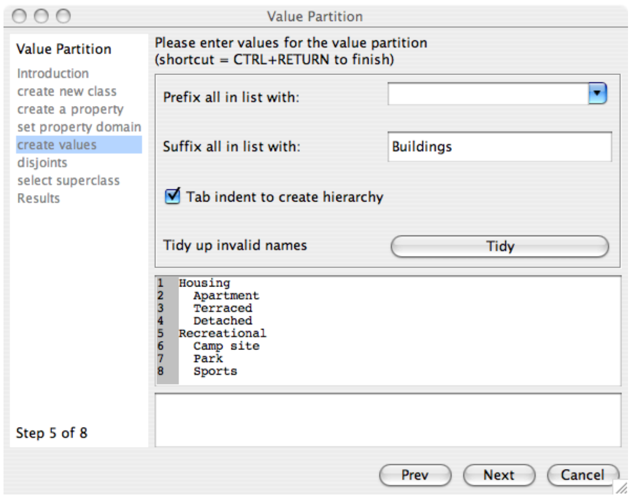


Figure 5.5: Protégé inline restriction editor

quickly or adding a new property and values to the skeleton ontology. Creating multiple subclasses quickly is a useful functionality when working on such ontologies as the Realraumanalyse one that has a lot of categories with similar names as only those parts of the category names need to be entered that are different and prefixes and suffixes can be added to create the full names of the categories. The second wizard is very useful when working on the skeleton ontology (fig. 5.6. A sub-tree in the skeleton ontology can be created quite quickly using a simple editor that uses indentation to specify the desired hierarchy and then performs all the necessary tasks such as creating the ontology concepts, the property setting the domain and range of the property and if desired making the concepts disjoint.

- Finally Protégé provides an integrated way to access *automated reasoners* such as Racer or FaCT (fig. 5.7). This makes it possible to use these reasoners to check the ontologies for consistency and also whether there are any concepts that are unsatisfiable due to their definitions. They can also be used to infer the hierarchy of the defined concepts and errors in the hierarchy can be used to find errors in the definitions.

While these may seem to be petty points the correctness of the modelled semantics is vital for all other functionality of the system and thus any support that a tool can offer to improve this correctness is useful.



Please enter values for the value partition
(shortcut = CTRL+RETURN to finish)

Prefix all in list with:

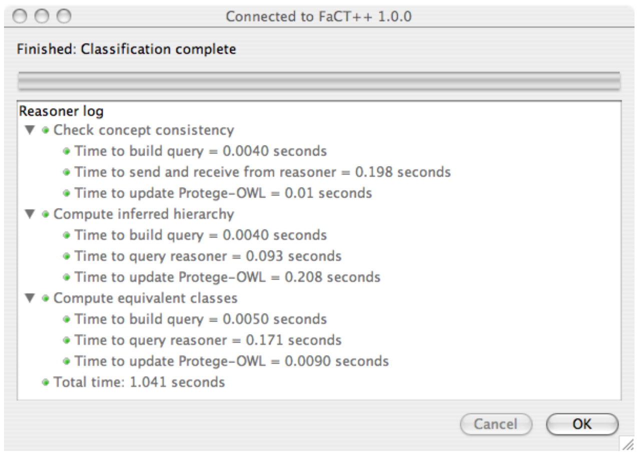
Suffix all in list with: Buildings

☑ Tab indent to create hierarchy

Tidy up invalid names

```
Housing
  Apartment
  Terraced
  Detached
Recreational
  Camp site
  Park
  Sports
```

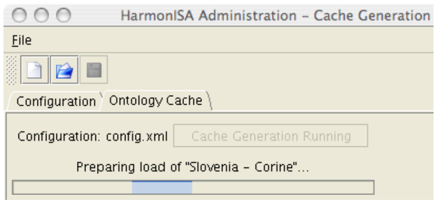
Figure 5.6: Protégé OWL wizard



Finished: Classification complete

```
Reasoner log
▼ Check concept consistency
    Time to build query = 0.0040 seconds
    Time to send and receive from reasoner = 0.198 seconds
    Time to update Protege-OWL = 0.01 seconds
▼ Compute inferred hierarchy
    Time to build query = 0.0040 seconds
    Time to query reasoner = 0.093 seconds
    Time to update Protege-OWL = 0.208 seconds
▼ Compute equivalent classes
    Time to build query = 0.0050 seconds
    Time to query reasoner = 0.171 seconds
    Time to update Protege-OWL = 0.0090 seconds
  Total time: 1.041 seconds
```

Figure 5.7: Protégé using the FaCT reasoner



Configuration: config.xml

Preparing load of "Slovenia – Corine"...

Figure 5.8: Administration tool creating the similarity mappings

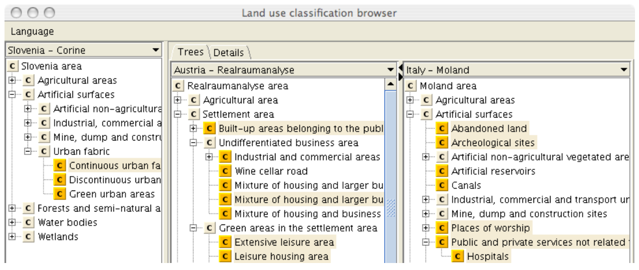
5.2.3 Semantic Similarity Calculation Library

When the required categories have been modelled in the ontologies the next step is to use the semantic similarity algorithm to create the integration mappings between the different ontologies. To do this an administrative application was developed that takes the different ontologies and uses the semantic similarity calculation library to calculate the mappings between the different ontologies (fig. 5.8).

The semantic similarity calculation library forms the core of the HarmonISA project. It contains the data structures and algorithms necessary to store the ontologies and the mappings between the different categories in the ontologies. first the ontologies are loaded from their OWL files and then the inferred is calculated hierarchy. To do this the OWL API and Pellet OWL Description Logics reasoner are used. The OWL API is used to load the OWL files into preliminary data structure on which the Pellet reasoner can work. The hierarchy that the Pellet reasoner infers is then stored in a data structure that was developed to make the similarity calculation and display of the similarity calculation results as simple as possible. This data structure is then fed into the actual similarity calculation algorithm.

The similarity calculation algorithm is an implementation of the algorithm described in section 4.2. It iterates through all ontologies, comparing each ontology to all of the others and storing the integration mapping results. For each ontology pair that is compared each category from the source ontology is compared to each category of the target ontology. Of course this is not very efficient but since the attempts at using data statistics to provide a heuristics to optimise the algorithm failed it is currently the fastest implementation. The similarity comparison for two categories leads to a number of comparison results, one for each of the properties that are compared. Each of the property comparison results contains further comparison results for the ranges of the properties that are compared. These comparison results are the first step in the similarity calculation. They are then fed into the similarity calculator that calculates the final similarity value between 0 and 1 out of the comparison results. The end result of this is a data structure containing the complete ontologies and for each concept in each ontology a list of concepts from the other ontologies that are most similar to it.

This whole process runs within the administrative application which after all calculations have been completed serialises the data structure so that it can then be used in the other applications that are part of the HarmonISA system.



Land use classification browser

Language

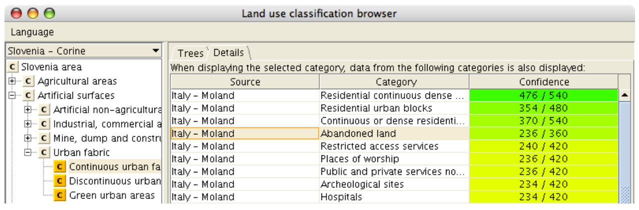
Trees | Details

```
Slovenia – Corine
Slovenia area
  Agricultural areas
  Artificial surfaces
    Artificial non-agricultura
    Industrial, commercial a
    Mine, dump and constru
    Urban fabric
      Continuous urban fa
      Discontinuous urban
      Green urban areas
  Forests and semi-natural a
  Water bodies
  Wetlands

Austria – Realraumanalyse
Realraumanalyse area
  Agricultural area
  Settlement area
    Built-up areas belonging to the publ
    Undifferentiated business area
      Industrial and commercial areas
      Wine cellar road
      Mixture of housing and larger bu
      Mixture of housing and larger bu
      Mixture of housing and business
    Green areas in the settlement area
      Extensive leisure area
      Leisure housing area

Italy – Moland
Moland area
  Agricultural areas
  Artificial surfaces
    Abandoned land
    Archeological sites
    Artificial non-agricultural vegetated are
    Artificial reservoirs
    Canals
    Industrial, commercial and transport ur
    Mine, dump and construction sites
    Places of worship
    Public and private services not related
      Hospitals
```

Figure 5.9: Tree view of the evaluation tool



Land use classification browser

Language

Trees | Details

```
Slovenia – Corine
Slovenia area
  Agricultural areas
  Artificial surfaces
    Artificial non-agricultura
    Industrial, commercial a
    Mine, dump and constru
    Urban fabric
      Continuous urban fa
      Discontinuous urban
      Green urban areas
```

When displaying the selected category, data from the following categories is also displayed:

Source	Category	Confidence
Italy – Moland	Residential continuous dense ...	476 / 540
Italy – Moland	Residential urban blocks	354 / 480
Italy – Moland	Continuous or dense residenti...	370 / 540
Italy – Moland	Abandoned land	236 / 360
Italy – Moland	Restricted access services	240 / 420
Italy – Moland	Places of worship	236 / 420
Italy – Moland	Public and private services no...	236 / 420
Italy – Moland	Archeological sites	234 / 420
Italy – Moland	Hospitals	234 / 420

Figure 5.10: Detail view of the evaluation tool

5.2.4 Similarity Mapping Evaluation

The serialised data structure is then loaded into the Ontology Browser that allows the ontology developer to both look at the ontology structure that has been created but also to inspect the mappings that have been calculated between the different concepts. It provides two views into the integration mappings. The first is a multi-tree view that makes it possible to quickly evaluate the mappings between concepts (fig. 5.9). On the left side is the tree structure of the currently selected ontology. On the right side are two tree views that can be set to other ontologies. When concepts are selected in the selected ontology on the left, then concepts in the other two ontologies that have been calculated as having the selected concept as the most similar concept are immediately highlighted. This makes it possible to quickly determine which concepts have been mapped to which concept and whether there are any erroneous mappings.

If such erroneous mappings have been found or there are mappings where it is unclear why the concepts have been mapped in that way then the second view comes into play. The detail view shows the same information as the tree view but in a list form (fig: 5.10). An additional functionality is that each of the mappings can be selected and the differences between the definitions of the two concepts can be analysed. This makes it possible to determine why a certain mapping has been calculated and whether this calculation is due to an incorrect definition of one of the two concepts.

After this evaluation of the mappings the decision is made whether a further iteration of modifying the definitions and calculating similarity is required or whether the results are satisfactory. If another iteration is needed then the process restarts with either re-evaluating the semantics of the concepts or directly with modifying the ontologies in Protégé. On the other hand if the results are satisfactory then they can be loaded into those applications of the HarmonISA project that provide the interface to the user.

5.2.5 HarmonISA Toolbox

The HarmonISA toolbox consists of a set of applications that are used in the development and maintenance of the knowledge used within the HarmonISA project. Two applications from the toolbox have already been mentioned namely the administrative application that creates the mappings and the evaluation application. In addition tools have been developed that cover the following functionalities

- *Documentation* of the semantics that have been encoded in the ontologies and also the mappings that have been created. These documents are used as documentation in the main application but also for external evaluation of the knowledge by domain experts.
- *Internationalisation* is an important part of the HarmonISA project since it is aimed at users in three different countries speaking three different languages. A tool has been developed to ease the communication between the internationalisation in the project and external translators.
- *Data translation* between the different catalogues. While the primary interface to the data is provided via the web application it is sometimes necessary to create a data set that contains the sum of all polygons from all source data sets translated into one LUC catalogue. This data set can then be used in other applications or for analysis.
- *Statistical analysis* of the data sets. This tool is is a leftover from the attempt to use statistical rankings to improve the performance of the similarity algorithm. While it is no longer used frequently it still provides interesting analytical results of the LUC data sets.

These tools form the scaffolding which is used to construct and maintain the primary application, the Harmonised Land-Use Viewer.

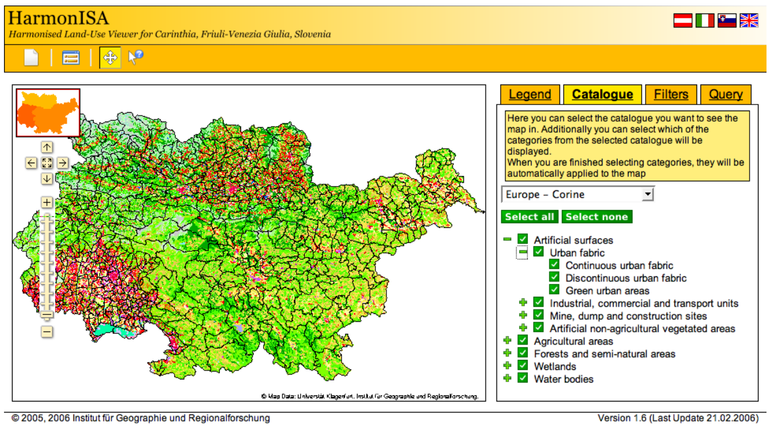
5.3 HarmonISA Harmonised Land-Use Viewer

The Harmonised Land-Use Viewer provides the primary interface onto the land-use data and similarity mappings that are the central product of the HarmonISA project. To reach the largest possible audience and to make it as easy as possible for everybody to access the results the viewer has been developed as a web application.

5.3.1 The User's View

When the user accesses the Harmonised Land-User Viewer[9] the application interface loads and the user is presented with the map of the region in the centre and a control panel on the right hand side (fig. 5.11). In the control panel on the right hand side the user can choose which LUC catalogue and from this catalogue which categories are to be displayed

[9]http://harmonisa.uni-klu.ac.at


HarmonISA
Harmonised Land-Use Viewer for Carinthia, Friuli-Venezia Giulia, Slovenia
Legend
Catalogue
Filters
Query
Select all
Select none

Here you can select the catalogue you want to see the map in. Additionally you can select which of the categories from the selected catalogue will be displayed.
When you are finished selecting categories, they will be automatically applied to the map

Europe – Corine

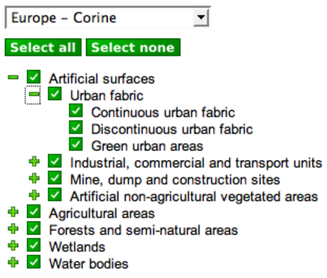
- ✓ Artificial surfaces
  - ✓ Urban fabric
    - ✓ Continuous urban fabric
    - ✓ Discontinuous urban fabric
    - ✓ Green urban areas
  - ✓ Industrial, commercial and transport units
  - ✓ Mine, dump and construction sites
  - ✓ Artificial non-agricultural vegetated areas
- ✓ Agricultural areas
- ✓ Forests and semi-natural areas
- ✓ Wetlands
- ✓ Water bodies

© Map Data: Universität Klagenfurt, Institut für Geographie und Regionalforschung.

© 2005, 2006 Institut für Geographie und Regionalforschung

Version 1.6 (Last Update 21.02.2006)

Figure 5.11: HarmonISA Web Application



Europe – Corine

- ✓ Artificial surfaces
  - ✓ Urban fabric
    - ✓ Continuous urban fabric
    - ✓ Discontinuous urban fabric
    - ✓ Green urban areas
  - ✓ Industrial, commercial and transport units
  - ✓ Mine, dump and construction sites
  - ✓ Artificial non-agricultural vegetated areas
- ✓ Agricultural areas
- ✓ Forests and semi-natural areas
- ✓ Wetlands
- ✓ Water bodies

Figure 5.12: HarmonISA Web Application category selection

in the map. The application has been designed in such a way that any changes that the user makes is immediately applied and the map area updated to reflect the new settings. The same holds true for the filter selection (fig. 5.12) which allows the user to select which catalogues to display and which to hide in the map. The user can then navigate the map using the controls on the left side of the map (fig. 5.13) that support panning and zooming. The user can also re-centre the map by clicking on the area to centre.

The system also provides access to similarity information. Either by using the query tool and clicking on the area for which the similarity information should be displayed or by using the catalogue viewer to access the complete similarity information for all categories. The similarity information is available in three levels of detail (fig. 5.14). Either all categories can be displayed, one category with its similar categories or the

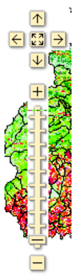
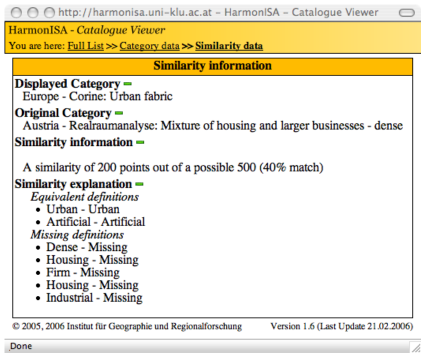


Figure 5.13: HarmonISA Web Application map controls



http://harmonisa.uni-klu.ac.at – HarmonISA – Catalogue Viewer

HarmonISA - *Catalogue Viewer*

You are here: Full List >> Category data >> **Similarity data**

Similarity information

Displayed Category
Europe - Corine: Urban fabric

Original Category
Austria - Realraumanalyse: Mixture of housing and larger businesses - dense

Similarity information

A similarity of 200 points out of a possible 500 (40% match)

Similarity explanation

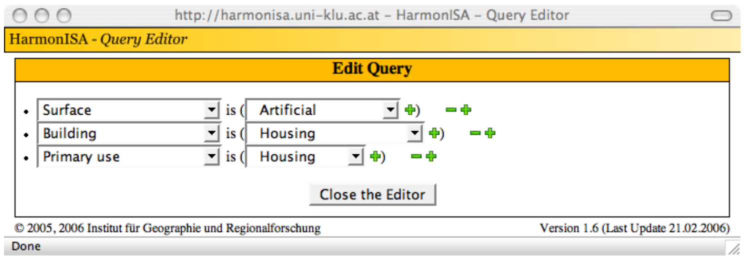
Equivalent definitions
- Urban - Urban
- Artificial - Artificial

Missing definitions
- Dense - Missing
- Housing - Missing
- Firm - Missing
- Housing - Missing
- Industrial - Missing

© 2005, 2006 Institut für Geographie und Regionalforschung Version 1.6 (Last Update 21.02.2006)

Done

Figure 5.14: HarmonISA Web Application catalogue viewer



http://harmonisa.uni-klu.ac.at – HarmonISA – Query Editor

HarmonISA - *Query Editor*

Edit Query

- Surface is (Artificial)
- Building is (Housing)
- Primary use is (Housing)

© 2005, 2006 Institut für Geographie und Regionalforschung Version 1.6 (Last Update 21.02.2006)

Done

Figure 5.15: HarmonISA Web Application semantic query definition

detailed comparison including explanation of the differences between two categories.

The final functionality that is provided by the application is performing a semantic query against the system. This functionality makes it possible to define a complex query that can then be run against the other LUC catalogues that are available in the system. The user uses the properties and values defined in the skeleton ontology to create a query (fig. 5.15) and the system then uses the semantic similarity algorithm to perform the query and displays the results in the map (fig. 5.16).

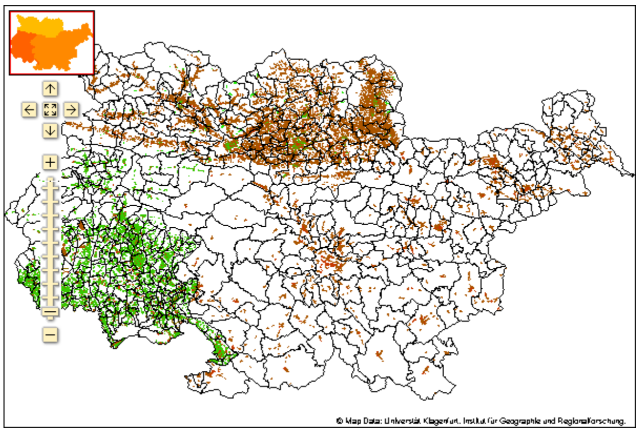

© Map Data: Universität Klagenfurt, Institut für Geographie und Regionalforschung

Figure 5.16: HarmonISA Web Application semantic query results

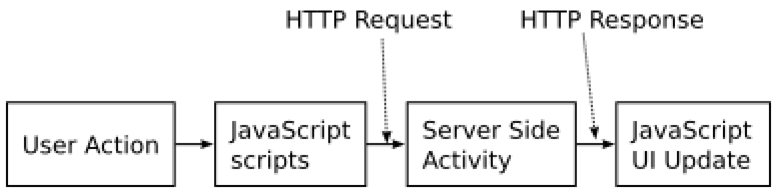


Figure 5.17: Workflow for the interaction between user interface and application

5.3.2 Under the Hood

The application itself is a combination of HTML[10] and CSS[11] for displaying the application interface, Java Servlets and Java Server Pages provide the server side functionality and JavaScript is used to make the interface interactive and also to glue the interface to the server side functionality. A typical activity follows the pattern described in figure 5.17. The user performs an action, for example selecting an additional category to display. Clicking on the box to select a category calls a JavaScript script that opens a HTTP[12] connection to the server and sends a request detailing the action the user has performed. The server receives the request with the action and applies it to the user's server side application state. In this case the category is selected and also all child categories are selected. The server then sends a response back to the JavaScript script including information on what was selected. The script then updates the user interface to reflect the server side state of the application.

What is important to note is that for every user a separate state is created. Thus at any time more than one instance of the application state will exist on the server side. Every time an action is performed and a request sent by the JavaScript to the server the server determines which user sent the request and fetches the correct application state. The actions are then performed on the state and sent back to the user. This makes it possible to write a generic handling code that only has to be instantiated once which in turn reduces the system requirements, while still supporting a large number of users.

Three main areas can be identified within the server side application

- *LUC catalogue handling* involves all actions such as selecting the catalogue and categories, applying filters and displaying similarity information.
- *Map rendering* handles the rendering of the maps that the user requests.
- *Querying* implements the map query and semantic query functionality of the system.

Most of the functionality is pretty standard for a web application but some areas deserve attention.

Internationalisation

The goal of the HarmonISA project is to allow users from the three countries involved in the ISA-Map project to access the land-use data of the complete region in the format

[10]HyperText Markup Language: The markup language for web pages, see http://www.w3c.org
[11]Cascading Style Sheets: A way of formatting web pages created using HTML
[12]HyperText Transport Protocol: The protocol used for requesting and sending web pages

and catalogue that they are accustomed to. This of course also involves the language of the interface. To enable this an internationalisation system has been integrated into the application that translates all parts of the user interface and also the different land-use categories into the desired language. Unfortunately automatic translation is not yet up to this task and thus the translations have to be done by translators and then statically included into the application.

Category selection and de-selection handling

The category selection and de-selection handling is an interesting area because it show-cases the situation where the technology used doesn't fully support the required functionality. As mentioned in section 4.2.1 some LUC categories are placed multiple times in the hierarchy of the catalogue. This means that the hierarchy is not a tree but a directed graph structure. While this is easily represented in the data structure that is used on the server side of the application since there a category can simply have multiple parents, it cannot be done in HTML which only supports tree structures. Thus the directed graph structure needs to be unfolded into a tree structure and this leads to the situation that some categories are present in more than one position in the HTML tree. Thus when the user selects or deselects one instance of the category in the tree then the other instances need to be selected or deselected as well. To do this each category on the server side stores a list of identifiers that identify the nodes in the HTML tree. This way the application knows which nodes in the HTML tree belong together and need to be selected/deselected together.

Map rendering and caching

To perform the rendering of the map the application uses functionality provided by the GeoTools library. While this library is very powerful and provides a lot of geoprocessing functionality the rendering can be very slow if the amount of data is very large. To combat this an elaborate caching strategy has been developed that builds on three pillars. The map is split up into a grid of map tiles, these map tiles can then be cached and the polygons in the map tiles can be pre-calculated to further increase performance. This leads to the process depicted in figure 5.18 when the map needs to be redrawn. First the user performs the action leading to the map needing to be redrawn, for example changing the zoom level. This action is transmitted to the server and the server determines the coordinates of the map tiles that are needed to display the desired map area. On the client side the JavaScript script updates the source information for the array of image elements that show the map tiles. This change induces the browser to request the new map tiles from the server[13]. Back at the server the map tile request contains the coordinates of the tile to be rendered. The application first checks the in memory cache whether this tile has already been rendered. If the image is found then it is immediately sent back to the browser and displayed. If it is not in the memory cache then the next level is the disk cache. Finally if the tile is also not found in the disk cache then it is newly rendered. This

[13]Another level of caching is actually included here, since the browser also caches images that have been loaded before.

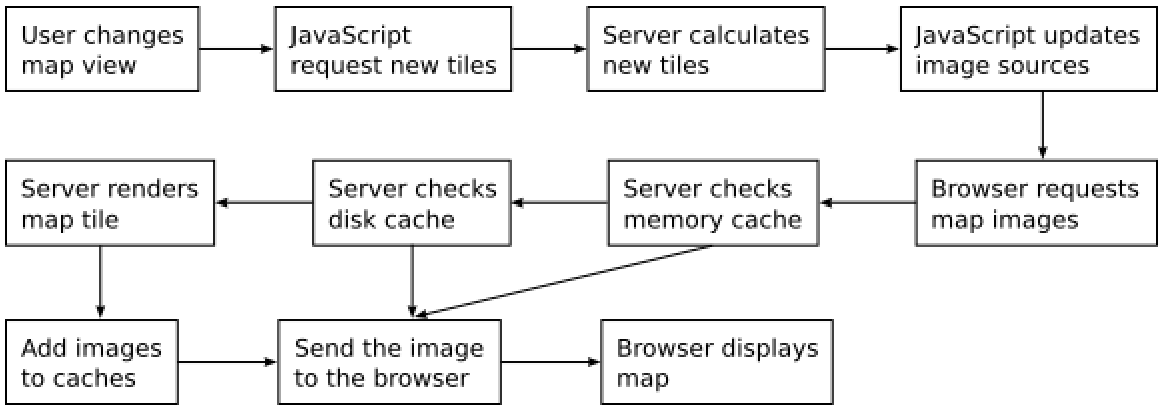

User changes map view
JavaScript request new tiles
Server calculates new tiles
JavaScript updates image sources
Browser requests map images
Server checks memory cache
Server checks disk cache
Server renders map tile
Add images to caches
Send the image to the browser
Browser displays map

Figure 5.18: Workflow for the process of rendering maps

rendering uses the pre-calculated set of polygons that are displayed in the map tile. The rendered image is then added to both caches and sent back to the browser and displayed. These three levels of caching together allow the system to reach a sufficiently high map display speed to provide a non-frustrating experience to the user.

What is important to note is that both caches are not specific to the user but shared by all users of the application. This means that it is necessary to not only identify the map tiles by the area that they cover but also by what catalogue was used to render the map tile and which categories were rendered from the catalogue in the tile. This is also necessary to avoid the browser displaying cached data when the catalogue is changed. The advantage of this is that the map tiles that were rendered for one user can also be used for all other users again increasing the efficiency of the system.

Semantic querying

The semantic querying functionality makes it possible for the user to query the LUC catalogues in the system using the semantic similarity algorithm. The way this works (fig. 5.19) is that first the user defines the query that should be performed. The query consists of a list of properties plus the ranges that these properties may have. This definition of the query is sent to the server where properties without ranges and other invalid parts of the query are filtered out. When the users starts the query the query definition on the server side is transformed into a virtual LUC category that has the query definition as its definition. This virtual LUC category is then compared to all the existing LUC catalogues using the semantic similarity algorithm. The results from this semantic comparison are then split into ten groups based on the similarity value with the exception of those comparisons that resulted in a similarity value of 0 which are discarded. The results are then added into a virtual LUC catalogue that is then automatically selected in the interface and the results of the query displayed.

There is one small difference between the way the semantic similarity algorithm is used when calculating the similarity between the LUC catalogues and when used for querying. The query definitions tend to be shorter than the ones in the definitions of the LUC categories and this leads to a lot of missing properties which reduces the clarity of the

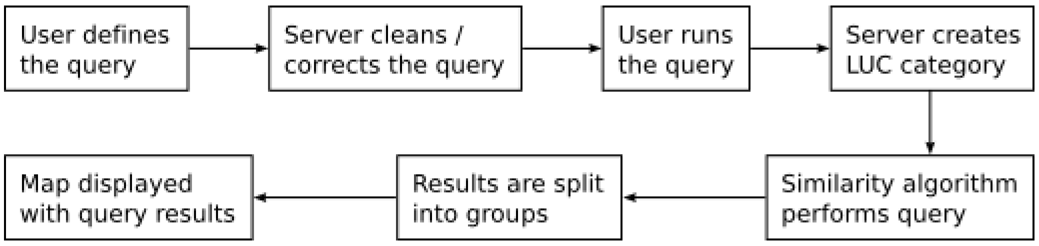

User defines the query
Server cleans / corrects the query
User runs the query
Server creates LUC category
Similarity algorithm performs query
Results are split into groups
Map displayed with query results

Figure 5.19: Workflow for semantic querying

query result. To combat this all properties that are not in the query definition are weighted by a factor of 0.2 to reduce their influence on the result. This factor has been determined empirically to deliver good query results and would have to be validated before it can be used in other domains.

6 Conclusions

Integration of data is a hard problem. There are a lot of obstacles that need to be overcome in order to integrate two or more data sources such as differing encodings, structures and scopes. If an attempt is made to also integrate the semantics of the data then the problem becomes even harder. More and larger obstacles must be overcome and in some cases cannot be fully solved at all. Semantic integration means having to deal with problems such as different scopes of concepts, different modelling principles, homonyms and synonyms. Automatic integration of semantics is even harder. While a human when performing a semantic integration can bring all his knowledge to the problem the computer is only aware of what it has been explicitly told. Thus it is necessary to create a very precise and explicit description of the semantics involved. Only based on such descriptions is it possible to perform automatic semantic integration of data and schemas.

This thesis introduced an algorithm to perform automatic semantic integration of ontologies. It is based on a methodology for ontology modelling that aims to maximise both the expressiveness of the ontology and also the ability to perform automated reasoning on the knowledge in the ontology. The ontology model and semantic similarity algorithm are based on a hybrid cognitive model that is constructed out of a combination of the feature model[Tver77] and network model[RadaEtAl89]. Since it is based on a cognitively valid model the results obtained by using it are also cognitively valid. Using this semantic similarity algorithm an application has been developed that automatically integrates land-use and land-cover data sets from the regions Carinthia, Friuli-Venezia Giulia and Slovenia. The combination of a cognitive model and a workable implementation what gives the algorithm described in this thesis an advantage over existing systems.

In order to validate the semantic similarity model and algorithm the results from applying them to the land-use domain have been evaluated by a number of experts from the land-use and geoinformation field. This evaluation highlighted some known and some unknown problem areas, but also some areas where the algorithm works very well. The results are consistently good enough to deploy the algorithm in a productive environment with the average number of correct mappings being between 95 and 100 percent. Of the strengths and weaknesses that were already known the greatest strength is at the same time one of the greatest weaknesses and that is the simplicity of the model. While this makes modelling very easy and the similarity calculation understandable it also severely restricts what can be expressed in the model. A further strength is that both are built on open standards which makes integration into other systems much easier as there are no proprietary hurdles to overcome.

A definite weakness of the algorithm is performance as the algorithm calculates all possible mappings between two ontologies and only then selects the best and also uses Description Logics reasoning in some areas. This leads to the whole system running in exponential time. For large and complex ontologies this makes the whole system basically unusable and is already problematic for the ontology sizes used in the development and evaluation of the algorithm. Optimisations are possible and could reduce the complexity from exponential to polynomial ($\mathrm{O}(N^5)$) which while a lot faster still precludes the use in an interactive environment. Nevertheless since the ontologies describing the data do not change often it is possible to precalculate the integration mapping and thus make it possible to deploy the system in a production environment.

One of the problems that derive from the simplicity of the model is that relations between concepts are not supported. This means that all or some of the knowledge that is inherent in the relations is lost when transferred into the hybrid model. Currently in such a situation where a relation between two concepts would be necessary an extra property is added to describe the contents of the relation. This makes it possible to model part of the knowledge of the relation, but not the actual relation itself. So for example for an alluvial forest a property LIES NEXT TO would be added and the value of the property set to RIVER. The problem is that the knowledge that the river value from the property is the same as the river in the LUC catalogue is only implicit within the names of the property and the property value and thus not available to the similarity algorithm. Then if a similar alluvial forest is defined in a different LUC catalogue the property is reused. To the similarity algorithm it now looks as if the two definitions are exactly the same, although in the original metadata they actually refer to the rivers in their respective LUC catalogues which might have differing definitions. Thus while this is a solution to the basic modelling deficiency it also creates further problems.

A more fundamental problem is that not everything can be reduced to reasoning on concepts. Many things either have continuous or fuzzy properties. For example in the Realraumanalyse catalogue the agricultural areas are classified based on relative amount of arable land and pastures in the area. This percentage is a continuous value between 0 and 100 with each agricultural category defining a lower and upper bound the percentage must have for an area to be classified into that category. Neither OWL nor the semantic similarity algorithm currently support reasoning on these kinds of properties and restrictions. The real problem here being that OWL does not support it, since while the algorithm can be changed this is not true for the OWL language.

Finally the model and algorithm suffer from the fact that the similarity algorithm only supports a very limited set of simple constructors. Each concept can only be defined using one set of conjunctions. It is not possible to specify that for parts of the definition there are multiple alternatives of how the concept is defined. For example in the relevant metadata the MIXED FOREST category is defined as a mixture of broad-leafed and coniferous trees, but has a second definition that includes sub-arctic coniferous forest with a tree height of under five meters. Since the model does not support these kinds of dual definitions one of the two must be chosen and in this case the more frequent one namely the mixture of broad-leafed and coniferous trees was chosen. Nevertheless this does not actually fully capture the semantics of the category. Another situation that cannot be modelled is when the definition contains exclusions such as the definition of DISCONTINUOUS URBAN

FABRIC for which the metadata says that it excludes areas that are used for holiday and vacation housing. Both problems can be solved by extending the model but the question then arises on what effect this has on the similarity algorithm.

Although the model and similarity algorithm are still very primitive and suffer from the shortcomings mentioned above the fact that they already produce results with more than 95% correctness implies that they can already be deployed in real world situations. Further study is necessary to find solutions for the remaining problems. Then the model and the similarity algorithm should be able to deal effectively and successfully with the knowledge involved in most real world situations.

List of Figures

List of Tables

Bibliography

[BaaderEtAl94] F. Baader, E. Franconi, B. Hollunder, B. Nebel, H. Profitlich. An empirical analysis of optimization techniques for terminological representation systems or: Making KRIS get a move on. In *Applied Artificial Intelligence. Special Issue on Knowledge Base Management*, 4: pages 109-132, 1994.

[BaaderNutt03] F. Baader and W. Nutt. Basic Description Logics. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi and P. Patel-Schneider, editors, *The Description Logic Handbook - Theory, Implementation and Applications*, Cambridge University Press, 2003.

[BechEtAl03] S. Bechhofer, Raphael Volz and Phillip Lord. Cooking the Semantic Web with the OWL API. In *The SemanticWeb - ISWC 2003*, LNCS 2870: pages 659-675, 2003.

[BenaEtAl05] B. Benatallah, M. Hacid, A. Leger, C. Rey and F. Toumani. On automating Web services discovery. In *VLDB Journal(2005)*, 14: pages 84-96, 2005.

[BernEtAl01] T. Berners-Lee, J. Hendler and O. Lassila. The Semantic Web In *Scientific American*, 284(5): pages 28-37, 2001.

[BrachLev85] Readings in Knowledge Representation. R. J. Brachman and H. J. Levesque, editors. Morgan Kaufmann, 1985.

[CastEtAl01] J. Gonzalez-Castillo, D. Trastour and C Bartolini. Description Logics for Matchmaking of Services. In *Proceedings of the KI-2001 Workshop on Applications of Description Logics*, 2001.

[CastEtAl03] S. Castano, A. Ferrara and S. Montanelli. H-MATCH: an Algorithm for Dynamically Matching Ontologies in Peer-based Systems. In *Proceedings of the First Workshop on Semantic Web and Databases (SWDB-03)*, 2003.

[CastEtAl04] S. Castano, A. Ferrara, S. Montanelli and G. Racca. Matching Techniques for Resource Discovery in Distributed System Using Heterogeneous Ontology Descriptions. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)*, 2: page 360, 2004

[CastEtAl05] S. Castano, A. Ferrara, S. Montanelli. Ontology-based Interoperability Services for Semantic Collaboration in Open Networked Systems. In *Proc. of the 1st Int. Conference on Interoperability of Enterprise Software and Applications (INTEROP-ESA 2005)*, 2005.

[DiNoEtAl03] T. Di Noia, E Di Sciascio, F M. Donini and M. Mongiello. Semantic Matchmaking in a P-2-P Electronic Marketplace. In *Proceedings of the 2003 ACM symposium on Applied computing*, pages 582-586, 2003.

[DoanEtAl02] , A. Doan, J. Madhavan, P. Domingos and A. Halevy. Learning to Map between Ontologies on the Semantic Web. In *Proceedings of the 11th international conference on World Wide Web*, pages 662-673, 2002.

[EuzeValt03] J. Euzenat and P. Valtchev. An integrative proximity measure for ontology alignment. In *Proceedings of the 1st Intl. Workshop on Semantic Integration*, pages 33-38, 2003.

[Gärd00] P. Gärdenfors. Conceptual Spaces: The Geometry of Thought. The MIT Press, 2000.

[Gärd04] P. Gärdenfors. How to Make the Semantic Web More Semantic. In *Formal Ontology in Information Systems, Proceedings of the Third International Conference (FOIS 2004)*, pages 17-34, 2004.

[GentMark97] D. Gentner and A. B. Markman. Structure Mapping in Analogy and Similarity. In *American Psychologist*, 52(1): pages 45-56, 1997.

[Gold94] R. L. Goldstone, Similarity, Interactive Activation and Mapping. In *Journal of Experimental Psychology: Learning, Memory and Cognition*, 20(1): 3-28, 1994.

[Grub93] T. R. Gruber. A translation approach to portable ontology specifications. In *Knowledge Acquisition*, 5(2), 1993.

[HaarMöll00] V. Haarslev and R. Möller. Consistency testing: The RACE experience. In *TABLEAUX 2000*, LNAI 1847, 2000.

[HaarMöll01a] V. Haarslev and R. Möller. High performance reasoning with very large knowledge bases: A practical case study. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 161-168, 2001.

[HaarMöll01b] V. Haarslev and R. Möller. RACER system description. In *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR 2001)*, LNAI 2083, pages 701-705, 2001.

[Haye79] P. J. Hayes. In D. Metzing, editor, *The logic of frames.* Walter de Gruyter and Co., pages 46 - 61, 1979.

[HollBaad91] B. Hollunder and F. Bader. Qualifying number restrictions in concept languages. In *Proceedings of KR-91*, pages 335-346, 1991.

[Horr98] I. Horrocks. Using an expressive Description Logic: FaCT or fiction? In *Proceedings of the 6th International Conference on Principles of Knowledge Representation an Reasoning (KR'98)*, pages 636-647, 1998.

[HorrEtAl00a] I. Horrocks, U. Sattler and S. Tobies. Practical Reasoning for Very Expressive Description Logics. In *Logic Journal of the IGPL*, 8(3): pages 239-264, 2000.

[HorrEtAl00b] I. Horrocks, U. Sattler and S. Tobies. Practical reasoning for expressive description logics. In *Proceedings of LPAR'99*, LNAI 1847, pages 67-71, 2000.

[HorrEtAl03] I. Horrocks, P. F. Patel-Schneider and F. van Harmelen. From SHIQ and RDF to OWL: The Making of a Web Ontology Language. In *Journal of Web Semantics*, 1(1): pages 7-26, 2003.

[HorrPate99] I. Horrocks and P. F. Patel-Schneider. Optimizing Description Logic subsumption. In *Journal of Logic and Computation*, 9(3): pages 267-293, 1999.

[HorrSatt02] I. Horrocks and U. Sattler. Optimised reasoning for SHIQ. In *Proceedings of ECAI 2002*, 2002.

[Klei01] M. Klein. Combining and relating ontologies: an analysis of problems and solutions. In *Workshop on Ontologies and Information Sharing IJCAI'01*, August 2001

[KlienEtAl04] E. Klien, U. Einspanier, M. Lutz and S. Hübner. An Architecture for Ontology-Based Discovery and Retrieval of Geographic Information. In *Proceedings of the 7th Conference on Geographic Information Science (AGILE 2004)*, 2004.

[LemmAren04] R. Lemmens and H. Arenas. Semantic Matchmaking in Geo Service Chains: Reasoning with a Location Ontology. In *Proceedings of the 15th International Workshop on Database and Expert Systems Applications (DEXA'04)*, pages 797 - 802, 2004.

[LiHorr04] L. Li and I. Horrocks. A Software Framework for Matchmaking Based on Semantic Web Technology. In *International Journal of Electronic Commerce*, 8(4): pages 39-60, 2004.

[MadhEtAl01] J. Madhavan, P. A. Bernstein and E. Rahm. Generic Schema Matching with Cupid. In *Proceedings of the 27th VLDB Conference*, 2001.

[MedjEtAl03] B. Medjahed, A. Bouguettaya, A. K. Elmagarmid. Composing Web services on the Semantic Web. In *The VLDB Journal The International Journal on Very Large Data Bases*, 12(4): pages 333-351, 2003.

[MaedEtAl02] A. Maedche, B. Motik, N. Silva and R. Volz. MAFRA - A MApping FRAmework for Distributed Ontologies. In *Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web : 13th International Conference, EKAW 2002*, page 235, 2002.

[MaedStaa02a] A Maedche, S Staab. Comparing Ontologies - Similarity Measures and a Comparison Study. In *Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web : 13th International Conference, EKAW 2002*, page 251, 2002.

[MaedStaa02b] A. Maedche and S. Staab. Measuring Similarity between Ontologies. In *Lecture Notes in Computer Science*, 2473: page 251, 2002.

[MorkBern04] P. Mork and P. A. Bernstein. Adapting a Generic Match Algorithm to Align Ontologies of Human Anatomy. In *Proceedings of the 20th International Conference on Data Engineering (ICDE'04)*, page 787, 2004.

[Miller95] G. A. Miller. WordNet: A Lexical Database for English. In *Communications of the ACM*, 38(11): 39-41, 1995.

[Mins81] Marvin Minskey. A framework for representing knowledge. In J. Haugeland, editor, *A framework for representing knowledge.* The MIT Press, 1981.

[NardBrac03] D. Nardi and R. J. Brachman. An Introduction to Description Logics. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider, editors, *The Description Logic Handbook - Theory, Implementation and Applications*, Cambridge University Press, 2003.

[NoyMuse00] N. F. Noy and M. A. Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 450-455, 2000.

[PaolEtAl02] M. Paolucci, T. Kawamura, T. R. Payne and K. Sycara. Semantic Matching of Web Services Capabilities. In *The Semantic Web - ISWC 2002: First International Semantic Web Conference*, pages 333, 2002.

[ParsSivr04] B. Parsia and E. Sivrin. Pellet: An OWL-DL Reasoner. In *ISWC 2004 Proceedings*, 2004

[Quil67] M. R. Quillian. Word concepts: A theory and simulation. In *Behavioral Science*, 12: 410-430, 1967.

[RadaEtAl89] R. Rada, H. Mili, E. Bicknell and M. Blettner. Development and application of a metric on semantic nets. In *IEEE transactions on systems, man and cybernetics*, 19(1): pages 17-30, 1989.

[RahmBern01] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema amatching. In *The VLDB Journal*, 10: pages 334-350, 2001.

[Rect03] A. L. Rector. Modularisation of domain ontologies implemented in description logics and related formalisms including owl. In *Proceedings of Knowledge Capture 2003*, pages 121-128. ACM, 2003.

[RussEtAl99] T. Russ, A. Valente, R. MacGregor and W. Swartout. Practical Experiences in Trading Off Ontology Usability and Reusability. In *Proceedings of the Knowledge Acquisition Workshop (KAW99)*, 1999.

[RodrEgen03] A. Rodríguez and M. Egenhofer. Determining Semantic Similarity Among Entity Classes from Different Ontologies. In *IEEE Transactions on Knowledge and Data Engineering*, 15(2): pages 442-456, 2003.

[Satt96] U. Sattler. A concept language extended with different kinds of transitive roles. In *20. Deutsche Jahrestagung für KI, Lecture Notes in Artificial Intelligence*, vol. 1137, 1996.

[Satt03] U. Sattler. Description Logics for Ontologies. In *Proceedings of the International Conference on Conceptual Structures (ICCS 2003)*, 2746, pages 96-117, Springer Verlag, 2003.

[Schi91] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proceedings of IJCAI-91*, pages 466-471, 1991.

[SchmSmol91] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. In *Artificial Intelligence*, 48(1): pages 1-26, 1991.

[Schw05] A. Schwering. Hybrid Model for Semantic Similarity Measurement. In *Lecture Notes in Computer Science*, Springer, 2005.

[SchwRaub05] A. Schwering and M. Raubal. Measuring Semantic Similarity between Geospatial Conceptual Regions. In *Lecture Notes in Computer Science*, Springer, 2005.

[Sege00] M. Seger. Rauminformationssystem Österreich - digitaler thematischer Datensatz des Staatsgebietes fertiggestellt. In Strobl, Blachke and Griesebner editors *Angewandte Geographische Informationsverarbeitung XII. Beiträge zuum AGIT-Symposium Salzburg*, pages 465-468, 2000

[SycaEtAl99] K. Sycara, M. Klusch, S. Widoff and J. Lu. Dynamic Service Matchmaking Among Agents in Open Information Environments. In *SIGMOD Record*, 28(1), 1999.

[TangEtAl03] H. Tangmunarunkit, S. Decker and C. Kesselman. Ontology-based Resource Matching in the Grid - The Grid meets the Semantic Web. In *Lecture Notes in Computer Science*, 2870: pages 706-721. 2003.

[Tobi01] S. Tobies. Complexity Results and Practical Algorithms for Logics in Knowledge Representation. In *PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen*, 2001.

[Tver77] A. Tversky. Features of Similarity. In *Psychological Review*, 84(4): pages 327-352, 1977.

[TverGati78] A. Tversky and I. Gati. In E. Rosch and B. Lloyd, editors, *Studies of Similarity. Cognition and Categorizsation*, pages 79-98, 1978.

[UschGrun96] M. Uschold and M. Gruniger. Ontologies: Principles, Methods and Applications. In *Knowledge Engineering Review*, 11(2), 1996.

[WillEtAl03] A. Williams, A. Padmanabhan and M. B. Blake. Local Consensus Ontologies for B2B-Oriented Service Composition. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 647-654, 2003.

Appendix A - Ontology Structures

Structure of the Realraumanalyse Ontology

- Settlement area
  - Primarily open development
    - Open development in general, undifferentiated
      - Uniform detached housing area
      - Different extensive peripheral uses
    - Splittered settlement areas, if not as specific objects
      - Uniform detached housing area
      - Different extensive peripheral uses
  - Primarily continuous urban areas
    - City centres
    - Other town centres
    - Rural settlement (along road)
    - Centres of not continuously built up settlements
    - Incomplete village along road
    - Chain villages
    - Mixture of housing and larger businesses - continuous
  - Other dense urban areas
    - Primarily densly built up urban settlement area with old buildings
      - Densly built-up urban settlement area
      - Dense urban area, primarily housing
        - Multi-storey apartment blocks
      - Dense urban area, multifunctional
      - Former village now part of the city
      - Mixture of housing and larger businesses - dense
      - Dominantly commercial areas
  - Undifferentiated business area
    - Mixture of housing and larger businesses - continuous
    - Mixture of housing and larger businesses - dense
    - Mixture of housing and business use
    - Wine cellar road
    - Industrial and commercial areas in general
      - Dominantly commercial areas
      - Industrialarea
        - Mining and dump site

    - Peripheral shopping centre
    - Gardenmarket, tree nursery
  - Built-up areas belonging to the public administration
    - Public parks
  - Green areas in the settlement area
    - Cemeteries
    - Public parks
    - Golf course
  - Sports facility
    - Golf course
  - Leisure housing area
  - Tourism and leisure facilities
  - Small gardens
  - Extensive leisure area
  - Historical buildings: chateau, monastery; ring-road buildings in Vienna
  - Transportation area
    - Motorways
    - Motorwaytunnel
    - Trainstation
    - Airport
  - Pasture also winter sports are, flat
  - Pasture also winter sports are, slight inclination
  - Pasture also winter sports are, moderate inclination
  - Pasture also winter sports are, steep
  - Pasture also winter sports are, mountain pasture elevation
  - Other sport and leisure area in the green area
    - Golf course
- Agricultural area
  - Arable land > 90%, flat
  - Arable land dominant with pastures > 0.50%, flat
  - Arable land - pastures mix 20% < 60%, flat
  - Pasture dominant with arable land > 0.50%, flat
  - Pasture > 90%, flat
  - Fruit plantations, arable-specialised crop complexes, flat
    - Vineyard, arable land - vineyard complex, flat
  - Specialised crops with arable / pasture areas, flat
  - Pasture also winter sports are, flat
  - Arable land > 90%, slight inclination
  - Arable land dominant with pastures > 0.50%, slight inclination
  - Arable land - pastures mix 20% < 60%, slight inclination
  - Pasture dominant with arable land > 0.50%, slight inclination
  - Pasture > 90%, slight inclination
  - Fruit plantations, arable-specialised crop complexes, slight inclination
    - Vineyard, arable land - vineyard complex, slight inclination
  - Specialised crops with arable / pasture areas, slight inclination
  - Pasture also winter sports are, slight inclination

Arable land > 90%, moderate inclination
Arable land dominant with pastures > 0.50%, moderate inclination
Arable land - pastures mix 20% < 60%, moderate inclination
Pasture dominant with arable land > 0.50%, moderate inclination
Pasture > 90%, moderate inclination
Fruit plantations, arable-specialised crop complexes, moderate inclination
Vineyard, arable land - vineyard complex, moderate inclination
Specialised crops with arable / pasture areas, moderate inclination
Pasture also winter sports are, moderate inclination
Arable land > 90%, steep
Arable land dominant with pastures > 0.50%, steep
Arable land - pastures mix 20% < 60%, steep
Pasture dominant with arable land > 0.50%, steep
Pasture > 90%, steep
Fruit plantations, arable-specialised crop complexes, steep
Vineyard, arable land - vineyard complex, steep
Specialised crops with arable / pasture areas, steep
Pasture also winter sports are, steep
Arable land > 90%, mountain pastures
Arable land dominant with pastures > 0.50%, mountain pastures
Arable land - pastures mix 20% < 60%, mountain pastures
Pasture dominant with arable land > 0.50%, mountain pastures
Pasture > 90%, mountain pasture
Fruit plantations, arable-specialised crop complexes, mountain pasture
Vineyard, arable land - vineyard complex, mountain pasture
Specialised crops with arable / pasture areas, mountain pasture
Pasture also winter sports are, mountain pasture elevation
Forests
Coniferous forest dominates
Black pine vegetation
Black pine with broad-leaved vegetation
Black pine forest with additional firs and larches
Black pine forest with firs and larches
Coniferous forest with rocky areas
Broad-leaved forest dominates
Broad-leaved forest with black pines (< 50%)
Broad-leaved forest with black pines, firs and larches
Mixed forest, coniferous forest dominates
Black pine with broad-leaved vegetation
Mixed and broad-leaved forest with rocky areas
Larger tracts of woods along rivers
Mixed forest, broad-leaved forest dominates
Broad-leaved forest with black pines (< 50%)
Broad-leaved forest with black pines, firs and larches
Sub-alpine - alpine Zone
Subalpine broad-leaved bushes, with partial tree vegetation

- Glacier
- Rubble and boulders
- Rocky terrain
- Alpine turf partially with trees
- Continuous alpine turf vegetation
  - Combination of knee timber with alpine turf
  - Alpine turf with trees and tree groups
- Continuous knee timber
- Knee timber partially with turf and rocks
- Pasture outside of the continuous settlement area
- Skipistes in the alpine Zone

Other areas
- Wetlands (Moors in agricultural areas), flat
- Wetlands (Moors in agricultural areas), slight inclination
- Wetlands (Moors in agricultural areas), moderate inclination
- Wetlands (Moors in agricultural areas), steep
- Wetlands (Moors in agricultural areas), eben
- Moor with tree vegetation
- Moors in the subalpine / alpine area
- Standing water body
- Water courses
- Glacial ski regions
- Skipistes in the alpine Zone
- Skipistes on alpine Turf
- Winter sport area in the woods and lower mountain pasture zone

Other non-built-up areas owned by the public administration
- Built-up areas belonging to the public administration
  - Public parks

Structure of the Corine Ontology

Artificial surfaces
- Urban fabric
  - Continuous urban fabric
  - Discontinuous urban fabric
  - Green urban areas
- Industrial, commercial and transport units
  - Industrial or commercial units
    - Mineral extraction sites
    - Dump sites
  - Road and rail networks and associated land
    - Port areas
    - Airports
- Mine, dump and construction sites
  - Mineral extraction sites

- Dump sites
- Construction sites

Artificial non-agricultural vegetated areas
- Green urban areas
- Sport and leisure facilities

Agricultural areas
- Arable land
  - Non-irrigated arable land
  - Permanently irrigated land
    - Rice fields
- Permanent crops
  - Fruit trees and berry plantations
    - Vineyards
  - Olive groves
- Pastures
- Heterogeneous agricultural areas
  - Annual crops associated with permanent crops
  - Complex cultivation patterns
  - Land principally occupied by agriculture, with significant areas of natural vegetation
  - Agro-forestry areas

Forests and semi-natural areas
- Forests
  - Agro-forestry areas
  - Broad-leafed forest
  - Coniferous forest
  - Mixed forest
- Scrub and/or herbaceous vegetation association
  - Natural graslands
  - Moors and heathland
  - Sclerophyllous vegetation
  - Transitional woodland shrub
- Open spaces with little or no vegetation
  - Beaches, dunes and sand plains
  - Bare rocks
  - Sparsely vegetated areas
  - Burnt areas
  - Glaciers and perpetual snow

Wetlands
- Inland wetlands
  - Inland marshes
  - Peatbogs
- Coastal wetlands
  - Salt marshes
  - Salines
  - Intertidal flats

Water bodies

- Inland waters
  - Water courses
  - Water bodies
- Marine waters
  - Coastal lagoons
  - Estuaries
  - Sea and ocean

Appendix B - Similarity mappings

Mappings Realraumanalyse to Corine

Evaluation key:
Mapping is correct: *ok*
Mapping is correct but reclassified by expert: *reclassified*
Mapping is wrong due to modelling error: *modelling*
Mapping is wrong due to modell shortcoming: *error*

Settlement area $\longrightarrow$ Artificial surfaces : *ok*

Other non-built-up areas owned by the public administration $\longrightarrow$ Artificial surfaces : *modelling*

Mixture of housing and larger businesses - dense $\longrightarrow$ Urban fabric : *ok*

Other dense urban areas $\longrightarrow$ Urban fabric : *modelling*

Primarily densly built up urban settlement area with old buildings $\longrightarrow$ Urban fabric : *modelling*

Densly built-up urban settlement area $\longrightarrow$ Urban fabric : *modelling*

Former village now part of the city $\longrightarrow$ Urban fabric : *ok*

Dense urban area, primarily housing $\longrightarrow$ Urban fabric : *ok*

Multi-storey apartment blocks $\longrightarrow$ Urban fabric : *ok*

Dense urban area, multifunctional $\longrightarrow$ Urban fabric : *ok*

Built-up areas belonging to the public administration $\longrightarrow$ Continuous urban fabric : *ok*

Mixture of housing and larger businesses - continuous $\longrightarrow$ Continuous urban fabric : *ok*

Extensive leisure area $\longrightarrow$ Continuous urban fabric : *modelling*

Leisure housing area $\longrightarrow$ Continuous urban fabric : *modelling*

Tourism and leisure facilities $\longrightarrow$ Continuous urban fabric : *modelling*

Historical buildings: chateau, monastery; ring-road buildings in Vienna $\longrightarrow$ Continuous urban fabric : *modelling*

Primarily continuous urban areas $\longrightarrow$ Continuous urban fabric : *ok*

Chain villages ⟶ Continuous urban fabric : *modelling*

Rural settlement (along road) ⟶ Continuous urban fabric : *ok*

Other town centres ⟶ Continuous urban fabric : *ok*

City centres ⟶ Continuous urban fabric : *ok*

Incomplete village along road ⟶ Continuous urban fabric : *modelling*

Centres of not continuously built up settlements ⟶ Continuous urban fabric : *ok*

Wine cellar road ⟶ Discontinuous urban fabric : *ok*

Mixture of housing and business use ⟶ Discontinuous urban fabric : *ok*

Open development in general, undifferentiated ⟶ Discontinuous urban fabric : *ok*

Uniform detached housing area ⟶ Discontinuous urban fabric : *ok*

Different extensive peripheral uses ⟶ Discontinuous urban fabric : *ok*

Splittered settlement areas, if not as specific objects ⟶ Discontinuous urban fabric : *ok*

Public parks ⟶ Green urban areas : *ok*

Cemeteries ⟶ Green urban areas : *ok*

Small gardens ⟶ Green urban areas : *ok*

Undifferentiated business area ⟶ Industrial, commercial and transport units : *ok*

Transportation area ⟶ Industrial, commercial and transport units : *ok*

Industrial and commercial areas in general ⟶ Industrial or commercial units : *ok*

Dominantly commercial areas ⟶ Industrial or commercial units : *ok*

Gardenmarket, tree nursery ⟶ Industrial or commercial units : *ok*

Industrialarea ⟶ Industrial or commercial units : *ok*

Mining and dump site ⟶ Industrial or commercial units : *modelling*

Peripheral shopping centre ⟶ Industrial or commercial units : *ok*

Motorways ⟶ Road and rail networks and associated land : *ok*

Trainstation ⟶ Road and rail networks and associated land : *ok*

Motorwaytunnel ⟶ Road and rail networks and associated land : *ok*

Airport ⟶ Airports : *ok*

Green areas in the settlement area ⟶ Artificial non-agricultural vegetated areas : *ok*

Golf course ⟶ Sport and leisure facilities : *ok*

Other sport and leisure area in the green area ⟶ Sport and leisure facilities : *ok*

Sports facility ⟶ Sport and leisure facilities : *ok*

Glacial ski regions ⟶ Sport and leisure facilities : *ok*

Skipistes on alpine Turf ⟶ Sport and leisure facilities : *ok*

Skipistes in the alpine Zone $\longrightarrow$ Sport and leisure facilities : *ok*

Winter sport area in the woods and lower mountain pasture zone $\longrightarrow$ Sport and leisure facilities : *ok*

Agricultural area $\longrightarrow$ Agricultural areas : *ok*

Arable land $>$ 90%, flat $\longrightarrow$ Non-irrigated arable land : *ok*

Arable land $>$ 90%, mountain pastures $\longrightarrow$ Non-irrigated arable land : *ok*

Arable land $>$ 90%, moderate inclination $\longrightarrow$ Non-irrigated arable land : *ok*

Arable land $>$ 90%, slight inclination $\longrightarrow$ Non-irrigated arable land : *ok*

Arable land $>$ 90%, steep $\longrightarrow$ Non-irrigated arable land : *ok*

Arable land dominant with pastures $>$ 10%, flat $\longrightarrow$ Non-irrigated arable land : *ok*

Arable land dominant with pastures $>$ 10%, mountain pastures $\longrightarrow$ Non-irrigated arable land : *ok*

Arable land dominant with pastures $>$ 10%, moderate inclination $\longrightarrow$ Non-irrigated arable land : *ok*

Arable land dominant with pastures $>$ 10%, slight inclination $\longrightarrow$ Non-irrigated arable land : *ok*

Arable land dominant with pastures $>$ 10%, steep $\longrightarrow$ Non-irrigated arable land : *ok*

Arable land - pastures mix 40% $<$ 60%, flat $\longrightarrow$ Non-irrigated arable land : *ok*

Arable land - pastures mix 40% $<$ 60%, mountain pastures $\longrightarrow$ Non-irrigated arable land : *ok*

Arable land - pastures mix 40% $<$ 60%, moderate inclination $\longrightarrow$ Non-irrigated arable land : *ok*

Arable land - pastures mix 40% $<$ 60%, slight inclination $\longrightarrow$ Non-irrigated arable land : *ok*

Arable land - pastures mix 40% $<$ 60%, steep $\longrightarrow$ Non-irrigated arable land : *ok*

Fruit plantations, arable-specialised crop complexes, flat $\longrightarrow$ Fruit trees and berry plantations : *ok*

Fruit plantations, arable-specialised crop complexes, mountain pasture $\longrightarrow$ Fruit trees and berry plantations : *ok*

Fruit plantations, arable-specialised crop complexes, moderate inclination $\longrightarrow$ Fruit trees and berry plantations : *ok*

Fruit plantations, arable-specialised crop complexes, slight inclination $\longrightarrow$ Fruit trees and berry plantations : *ok*

Fruit plantations, arable-specialised crop complexes, steep $\longrightarrow$ Fruit trees and berry plantations : *ok*

Vineyard, arable land - vineyard complex, flat $\longrightarrow$ Vineyards : *ok*

Vineyard, arable land - vineyard complex, mountain pasture $\longrightarrow$ Vineyards : *ok*

Vineyard, arable land - vineyard complex, moderate inclination $\longrightarrow$ Vineyards : *ok*

Vineyard, arable land - vineyard complex, slight inclination $\longrightarrow$ Vineyards : *ok*

Vineyard, arable land - vineyard complex, steep $\longrightarrow$ Vineyards : *ok*

Pasture $>$ 90%, flat $\longrightarrow$ Pastures : *ok*

Pasture $>$ 90%, mountain pasture $\longrightarrow$ Pastures : *ok*

Pasture $>$ 90%, moderate inclination $\longrightarrow$ Pastures : *ok*

Pasture $>$ 90%, slight inclination $\longrightarrow$ Pastures : *ok*

Pasture $>$ 90%, steep $\longrightarrow$ Pastures : *ok*

Pasture dominant with arable land $>$ 10%, flat $\longrightarrow$ Pastures : *ok*

Pasture dominant with arable land $>$ 10%, mountain pastures $\longrightarrow$ Pastures : *ok*

Pasture dominant with arable land $>$ 10%, moderate inclination $\longrightarrow$ Pastures : *ok*

Pasture dominant with arable land $>$ 10%, slight inclination $\longrightarrow$ Pastures : *ok*

Pasture dominant with arable land $>$ 10%, steep $\longrightarrow$ Pastures : *ok*

Pasture also winter sports are, flat $\longrightarrow$ Land principally occupied by agriculture, with significant areas of natural vegetation : *ok*

Pasture also winter sports are, mountain pasture elevation $\longrightarrow$ Land principally occupied by agriculture, with significant areas of natural vegetation : *ok*

Pasture also winter sports are, moderate inclination $\longrightarrow$ Land principally occupied by agriculture, with significant areas of natural vegetation : *ok*

Pasture also winter sports are, slight inclination $\longrightarrow$ Land principally occupied by agriculture, with significant areas of natural vegetation : *ok*

Pasture also winter sports are, steep $\longrightarrow$ Land principally occupied by agriculture, with significant areas of natural vegetation : *ok*

Specialised crops with arable / pasture areas, flat $\longrightarrow$ Land principally occupied by agriculture, with significant areas of natural vegetation : *ok*

Specialised crops with arable / pasture areas, mountain pasture $\longrightarrow$ Land principally occupied by agriculture, with significant areas of natural vegetation : *ok*

Specialised crops with arable / pasture areas, moderate inclination $\longrightarrow$ Land principally occupied by agriculture, with significant areas of natural vegetation : *ok*

Specialised crops with arable / pasture areas, slight inclination $\longrightarrow$ Land principally occupied by agriculture, with significant areas of natural vegetation : *ok*

Specialised crops with arable / pasture areas, steep $\longrightarrow$ Land principally occupied by agriculture, with significant areas of natural vegetation : *ok*

Sub-alpine - alpine Zone $\longrightarrow$ Forests and semi-natural areas : *reclassified*

Forests $\longrightarrow$ Forests : *reclassified*

Larger tracts of woods along rivers $\longrightarrow$ Forests : *ok*

Broad-leaved forest dominates $\longrightarrow$ Broad-leafed forest : *ok*

Broad-leaved forest with black pines ($< 50\%$) $\longrightarrow$ Broad-leafed forest : *ok*

Broad-leaved forest with black pines, firs and larches $\longrightarrow$ Broad-leafed forest : *reclassified/modelling*

Black pine with broad-leaved vegetation $\longrightarrow$ Coniferous forest : *reclassified/modelling*

Coniferous forest dominates $\longrightarrow$ Coniferous forest : *ok*

Coniferous forest with rocky areas $\longrightarrow$ Coniferous forest : *ok*

Black pine vegetation $\longrightarrow$ Coniferous forest : *ok*

Black pine forest with firs and larches $\longrightarrow$ Coniferous forest : *ok*

Black pine forest with additional firs and larches $\longrightarrow$ Coniferous forest : *ok*

Mixed and broad-leaved forest with rocky areas $\longrightarrow$ Mixed forest : *ok*

Mixed forest, broad-leaved forest dominates $\longrightarrow$ Mixed forest : *ok*

Mixed forest, coniferous forest dominates $\longrightarrow$ Mixed forest : *ok*

Continuous alpine turf vegetation $\longrightarrow$ Scrub and/or herbaceous vegetation association : *reclassified*

Alpine turf with trees and tree groups $\longrightarrow$ Scrub and/or herbaceous vegetation association : *ok*

Combination of knee timber with alpine turf $\longrightarrow$ Scrub and/or herbaceous vegetation association : *ok*

Subalpine broad-leaved bushes, with partial tree vegetation $\longrightarrow$ Scrub and/or herbaceous vegetation association : *ok*

Continuous knee timber $\longrightarrow$ Scrub and/or herbaceous vegetation association : *ok*

Pasture outside of the continuous settlement area $\longrightarrow$ Scrub and/or herbaceous vegetation association : *reclassified*

Alpine turf partially with trees $\longrightarrow$ Bare rocks : *reclassified*

Rocky terrain $\longrightarrow$ Bare rocks : *ok*

Knee timber partially with turf and rocks $\longrightarrow$ Bare rocks : *error*

Rubble and boulders $\longrightarrow$ Sparsely vegetated areas : *modelling*

Glacier $\longrightarrow$ Glaciers and perpetual snow : *ok*

Wetlands (Moors in agricultural areas), flat $\longrightarrow$ Wetlands : *ok*

Wetlands (Moors in agricultural areas), eben $\longrightarrow$ Wetlands : *ok*

Wetlands (Moors in agricultural areas), moderate inclination $\longrightarrow$ Wetlands : *ok*

Wetlands (Moors in agricultural areas), slight inclination $\longrightarrow$ Wetlands : *ok*

Wetlands (Moors in agricultural areas), steep $\longrightarrow$ Wetlands : *ok*

Moors in the subalpine / alpine area $\longrightarrow$ Wetlands : *ok*

Moor with tree vegetation ⟶ Wetlands : *ok*

Other areas ⟶ Water bodies : *ok*

Water courses ⟶ Water courses : *ok*

Standing water body ⟶ Water bodies : *ok*

Mappings Corine to Realraumanalyse

Evaluation key:
Mapping is correct: *ok*
Mapping is correct but reclassified by expert: *reclassified*
Mapping is wrong due to modelling error: *modelling*
Mapping is wrong due to modell shortcoming: *error*

Artificial surfaces ⟶ Settlement area : *ok*

Artificial non-agricultural vegetated areas ⟶ Settlement area : *modelling*

Industrial, commercial and transport units ⟶ Settlement area : *ok*

Mine, dump and construction sites ⟶ Settlement area : *ok*

Construction sites ⟶ Settlement area : *ok*

Discontinuous urban fabric ⟶ Open development in general, undifferentiated : *ok*

Continuous urban fabric ⟶ Primarily continuous urban areas : *ok*

Urban fabric ⟶ Other dense urban areas : *ok*

Green urban areas ⟶ Dense urban area, multifunctional : *modelling*

Industrial or commercial units ⟶ Industrial and commercial areas in general : *ok*

Dump sites ⟶ Mining and dump site : *ok*

Mineral extraction sites ⟶ Mining and dump site : *ok*

Port areas ⟶ Transportation area : *ok*

Road and rail networks and associated land ⟶ Trainstation : *ok*

Airports ⟶ Airport : *ok*

Sport and leisure facilities ⟶ Other sport and leisure area in the green area : *ok*

Agricultural areas ⟶ Agricultural area : *ok*

Arable land ⟶ Agricultural area : *reclassified*

Heterogeneous agricultural areas ⟶ Agricultural area : *reclassified*

Complex cultivation patterns ⟶ Agricultural area : *reclassified*

Land principally occupied by agriculture, with significant areas of natural vegetation ⟶ Agricultural area : *reclassified*

Pastures $\longrightarrow$ Agricultural area : *reclassified*

Permanent crops $\longrightarrow$ Agricultural area : *reclassified*

Non-irrigated arable land $\longrightarrow$ Arable land $>$ 90%, flat : *ok*

Permanently irrigated land $\longrightarrow$ Arable land $>$ 90%, flat : *ok*

Rice fields $\longrightarrow$ Arable land $>$ 90%, flat : *ok*

Annual crops associated with permanent crops $\longrightarrow$ Fruit plantations, arable-specialised crop complexes, flat : *reclassified*

Fruit trees and berry plantations $\longrightarrow$ Fruit plantations, arable-specialised crop complexes, flat : *ok*

Vineyards $\longrightarrow$ Vineyard, arable land - vineyard complex, flat : *ok*

Olive groves $\longrightarrow$ Vineyard, arable land - vineyard complex, flat : *reclassified*

Agro-forestry areas $\longrightarrow$ Forests : *ok*

Forests $\longrightarrow$ Forests : *ok*

Coniferous forest $\longrightarrow$ Coniferous forest dominates : *ok*

Broad-leafed forest $\longrightarrow$ Broad-leaved forest dominates : *ok*

Mixed forest $\longrightarrow$ Broad-leaved forest dominates : *ok*

Forests and semi-natural areas $\longrightarrow$ Sub-alpine - alpine Zone : *reclassified*

Open spaces with little or no vegetation $\longrightarrow$ Sub-alpine - alpine Zone : *reclassified*

Beaches, dunes and sand plains $\longrightarrow$ Sub-alpine - alpine Zone : *reclassified*

Burnt areas $\longrightarrow$ Sub-alpine - alpine Zone : *reclassified*

Sparsely vegetated areas $\longrightarrow$ Sub-alpine - alpine Zone : *ok*

Moors and heathland $\longrightarrow$ Sub-alpine - alpine Zone : *ok*

Scrub and/or herbaceous vegetation association $\longrightarrow$ Subalpine broad-leaved bushes, with partial tree vegetation : *reclassified*

Natural graslands $\longrightarrow$ Subalpine broad-leaved bushes, with partial tree vegetation : *reclassified*

Sclerophyllous vegetation $\longrightarrow$ Subalpine broad-leaved bushes, with partial tree vegetation : *reclassified*

Transitional woodland shrub $\longrightarrow$ Subalpine broad-leaved bushes, with partial tree vegetation : *reclassified*

Glaciers and perpetual snow $\longrightarrow$ Glacier : *ok*

Bare rocks $\longrightarrow$ Rocky terrain : *ok*

Water bodies $\longrightarrow$ Other areas : *reclassified*

Inland waters $\longrightarrow$ Other areas : *reclassified*

Marine waters $\longrightarrow$ Other areas : *reclassified*

Coastal lagoons $\longrightarrow$ Other areas : *reclassified*

Estuaries $\longrightarrow$ Other areas : *reclassified*

Sea and ocean $\longrightarrow$ Other areas : *reclassified*

Wetlands $\longrightarrow$ Other areas : *reclassified*

Coastal wetlands $\longrightarrow$ Other areas : *reclassified*

Intertidal flats $\longrightarrow$ Other areas : *reclassified*

Salines $\longrightarrow$ Other areas : *reclassified*

Salt marshes $\longrightarrow$ Other areas : *reclassified*

Inland wetlands $\longrightarrow$ Other areas : *reclassified*

Inland marshes $\longrightarrow$ Other areas : *reclassified*

Peatbogs $\longrightarrow$ Other areas : *reclassified*

Water bodies $\longrightarrow$ Standing water body : *ok*

Water courses $\longrightarrow$ Water courses : *ok*