MASARYKOVA UNIVERZITA V BRNĚ
FAKULTA INFORMATIKY

Ontology Learning

DIPLOMA THESIS

Vít Nováček

Brno, winter 2005

Declaration

Hereby I declare, that this paper is my original authorial work, which I have worked out by myself. All sources, references and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Advisor: Aleš Horák, Ph.D.

Thanks

I would like to thank both my former and latter thesis advisors RNDr. Pavel Smrž, Ph.D. and Aleš Horák, Ph.D. for their expert support and scholarly supervisory. The research and tools presented here are parts of a broader project run and funded by Czech Academy of Sciences under the national research project 1ET100300419. I have appreciated this support very much and therefore I want to give a word of thanks also to all responsible people involved — especially doc. PhDr. Karel Pala, CSc. from Faculty of Informatics, Masaryk University Brno and Ing. Julius Stuller, CSc. from Czech Academy of Sciences.

Keywords

artificial intelligence, natural language processing, ontology, ontology acquisition, knowledge representation, text mining, knowledge extraction, uncertainty representation

Abstract

Ontology learning is one of the essential topics in the scope of an important area of current computer science and artificial intelligence – the upcoming Semantic Web. As the Semantic Web idea comprises semantically annotated descendant of the current world wide web and related tools and resources, the need of vast and reliable knowledge repositories is obvious. Ontologies present well defined, straightforward and standardised form of these repositories. There are many possible utilisations of ontologies – from automatic annotation of web resources to domain representation and reasoning tasks. However, the ontology creation process is very expensive, time-consuming and unobjective when performed manually. So a framework for automatic acquisition of ontologies would be very advantageous. In this work we present such a framework called OLE (an acronym for Ontology LEarning) and current results of its application. The main relevant topics, state of the art methods and techniques related to ontology acquisition are discussed as a part of theoretical background for the presentation of the OLE framework and respective results. Moreover, we describe also preliminary results of progressive research in the area of uncertain fuzzy ontology representation that will provide us with natural and reasonable instruments for dealing with inconsistencies in empiric data as well as for reasoning. Main future milestones of the ongoing research are debated as well.

Contents

Chapter 1

Introduction

The term *ontology* in the computer science is not strictly bound to the Semantic Web[1], but it has become widely discussed and examined in the very scope of this popular and progressive area. World Wide Web Consortium non-formally defines the Semantic Web as an activity that is "bringing the web to its full potential" in the meaning of extending its current capabilities by fully incorporated machine readable information and automated services in general (as stated in [30]). The content of such a web will be comprehensible not only by humans, but even by autonomous artificial agents.

However, ontologies can offer support for semantic annotation of web pages as well as present representation background for any domain of human knowledge in general. Their importance is not limited only to Semantic Web then, but is related to many areas of cognitive science and artificial intelligence in general. That is why extensive and universal ontology acquisition forms a bottleneck in many application areas from the above fields. Thus it is obvious that it is more than worthwhile to pursue development of a platform able to perform this task automatically, efficiently and reliably.

But what does the *ontology* term mean exactly? In the field of computer science, an ontology is usually understood as a formal and machine readable representation of conceptual sets, stratified in classes and including relations among particular concepts and their classes. Ontologies are able to provide a comprehensive representation of information related to a particular subdomain of human knowledge.

Basic approach to the ontology acquisition is a manual definition of domain conceptualisation. This task is usually performed by a group of domain experts. Various elaborated tools support the work; the most popular ones are Protégé, WebODE and OntoEdit. A comprehensive survey of such ontology engineering frameworks can be found in [4].

Manual creation of ontologies presents a tedious work, is error-prone

1. We can track notions akin to ontology purpose in machine-oriented knowledge representation about thirty years ago, when considering for example John Sowa's conceptual graphs [40].

and the results are often too subjective. Moreover, it is infeasible to organise a group of experts for each possible domain. This led to the idea of automatic extraction of ontologies from available resources.

In this work we devise OLE — a new complex language–independent platform for automatic and empiric ontology acquisition from natural language resources. In contrast to other ontology-learning systems that are currently available, OLE can be characterised by the modular architecture enabling integration and comparison of various methods of the automatic acquisition of semantic relations and respective ontology concepts. Virtually any method of automated knowledge acquisition can be employed as an independent part of the extraction module.

We introduce the architecture of the framework and discuss the methodology of the employed empiric approach. OLE enables creating the core taxonomy of an ontology subdomain in the bottom-up manner, from ontologies with a very simple structure to more complex ones, in a continual iterative process. It is also able to extend, refine and update ontologies with respect to new data. A miniontology for each input resource is created first. It consists of concepts and classes gained from the given resource. The miniontologies are integrated into the current ontology on the fly. The process of ontology merging and alignment shall embody an application of uncertainty representation methods.

The structure of the work is organised as follows. We analyse the problems embraced within the automatic ontology acquisition and application tasks in a representative survey in the theoretical parts of the work. These problems can be divided basically in the following general areas (though they may not be strictly disjunctive):

- ontology definition and representation
- extraction of concepts, relations and properties
- extraction or identification of ontological individual instances
- merging of ontologies
- efficient and meaningful reasoning

We give an overview of the selected principles and state of the art techniques for the above areas in Chapter 2, 3 and 5. Approaches selected for OLE project are depicted and discussed in the respective thematically dedicated Chapter 4 and 5. Concrete structure of the framework and more or less technical details are conveyed mainly in Chapter 6. Chapter 7 describes preliminary evaluation and interpretation of the results that have

been achieved with OLE application so far. Chapter 8 concludes the work and outlines the directions for future research on OLE. Appendices of the work offer examples of system application and a transcript of constructs that allow fuzzy extension of OWL.

Chapter 2

Ontology Formalisations and Theoretical Background

Before we turn our attention to ontology acquisition itself, ontology definitions and formal background should be discussed properly. In the following sections there is a list of ontology definitions, mainly with respect to computer science domain. Further we sketch possible ontology representations and utilisations. Eventually we discuss the natural connection of ontologies with the Semantic Web.

2.1 Formal Definitions of Ontology and Related Concepts

The ontology term has arisen from the philosophy field originally. According to glossary at `http://www.atf.org.au/papers/glossary.asp`, *ontology* means "branch of philosophy concerned with the study of being, of reality in its most fundamental and comprehensive forms" then. We can find many other similar philosophical definitions of ontology, but this one is fully sufficient for our needs.

Various computer science ontology definitions are brought out in the sections below. As we can see, the notion is quite similar to the philosophical one though it is much more specifically oriented and the ontology term refers rather to an object than to a field or branch. Its purpose shifts from study to representation of world or its thematically restricted part that is called *domain* most usually. According to very often cited short and general definition by Gruber [22], an ontology is an explicit specification of a conceptualisation of a given domain.

Read on in the more explicit definitions below. Note that the definition's list also reflects changing formal restrictions of knowledge representation from very vague to precise specifications influenced by the paradigm of ontology utilisation (these have been mainly expert systems, frame-based knowledge representations, support for artificial agents' cooperation and presently the Semantic Web).

2.1.1 Ontology as Vocabulary and Restrictions

Fikes presents sort of informal, yet a reasonable ontology definition in [17]. According to the source, ontologies are generally considered to provide definitions for the vocabulary used to represent knowledge. There should also be included sentences that restrict possible interpretations of undefined symbols. Ontologies must therefore contain both sentences like that and definitions. Fikes gives also a model for ontology role in a general declarative knowledge representation language. Such a language provides a syntax, a set of inference rules, a vocabulary of non-logical symbols, and the above-mentioned ontology sentences that restrict the acceptable interpretations of the symbols in the vocabulary. This model is mentioned in the scope of frame-based knowledge representations and KIF [20] format that had been very popular before the introduction and florescence of Semantic Web.

2.1.2 John Sowa's Complex Definition

John Sowa offers more elaborated and complex definition that naturally connects the ontology interpretation in philosophy with ontology as a descriptive tool in computer science. He specifies the following notions of the term (adopted from [41]):

1. **Generic and informal ontology definition**: The subject of ontology is the study of the categories of things that exist or may exist in some domain. The product of such a study, called an ontology, is a catalogue of the types of things that are assumed to exist in a domain of interest. The types are either undefined or defined only by statements in a natural language.

2. **Formal ontology definition**: A formal ontology is specified by a collection of names for concept and relation types organised in a partial ordering by the type-subtype relation. Formal ontologies are further distinguished by the way the subtypes differ from their supertypes:

    - Axiomatised ontologies: These distinguish subtypes by axioms and definitions stated in a formal language, such as logic or some computer-oriented notation that can be translated to logic.
    - Prototype-based ontologies: Subtypes are distinguished by comparisons with a typical member or prototype for each subtype.

2.1.3 Semantic Web Reflections and OLE Ontologies

Semantic Web efforts build on the definitions presented above — they do not bring brand new specifications of what an ontology is. They rather formalise the requirements posed on the languages for ontology representation. Such initiatives vary from a design of logical formalisms that allow to encode and process knowledge efficiently by autonomous software agents to proposals and development of the ontology representation languages that satisfy the requirements of these logics.

W3 Consortium (see `http://www.w3.org`) has become the key institution that steers almost all Semantic Web formalisation and proposal activities. It gives widely accepted standards for Semantic Web ontologies through the proper OWL (Web Ontology Language) specification (look at `http://www.w3.org/2004/OWL/` for recent details on OWL proposal, reference case–studies and so forth).

The consortium presents an implicit ontology definition somehow by the OWL recommendations, which we briefly recall here and develop in detail in Section 2.2. Perhaps the most crucial construct of OWL is **class** that roughly corresponds to *concept* in the meaning of representation of an entity existing in the relevant domain[1]. Entity defined by a class usually represents an abstract collection of other objects (e. g. the concept of *mammal* comprises objects like *carnivore*, *herbivore* or even *human* etc.). On the other hand, **individual** is a construct that represents concrete entity identifiable in the domain — an instance of a class (e. g. the *Tom the cat* individual is a concrete instance concept of the *cat* class concept). The third construct of the OWL triumvirate is **property**. Properties define attributes and/or relations of concepts (classes or individuals) in general.

Definitions of ontology content and its axiomatic restrictions are all represented using the three basic OWL constructs[2]. For the purpose of OLE ontologies we adopt structure similar to the one described in the paragraph above. However, we incorporate it into a more universal framework in order to tackle uncertain knowledge (yet with possible extensions that use intrinsic elements of OWL syntactic structure — see Section 5.4 for details).

1. We give up on a precise definition of *concept* presented for example in [31], because there has not been a reasonable consensus in the scope of formal semantics. We use the terms like concept, class, individual etc. rather in their technical meaning as proposed here and in Section 2.2.
2. Such an implicit ontology representation corresponds mostly to axiomatised Sowa's definition.

2.2 Ontology Languages and Classification

One of the first significant efforts in the area of machine readable knowledge representation is related to the establishment of the common communication framework between different software agents (according to Sowa in [41]). The knowledge sharing purpose is common for almost all ontology utilisations that have been presented among the divergent knowledge engineering community so far. However, the ontology representations adopted have been quite different. We sketch major ontology representation languages, proposed ranges of utilisation and formal classification of ontology types in Section 2.2.1 and Section 2.2.2. In Section 2.2.3 we introduce our way of ontology representation and empiric classification then.

2.2.1 Main Knowledge Representation Standards

It seems to be quite reasonable to use matured database technology (mainly relational or object-oriented database concepts, as mentioned in [39]) if we want to represent knowledge. Although there can be spotted similarities between ontologies and databases (see [32]), generic database schemata suffer from lack of tools for coherent and straightforward expression of conceptual relations and logical constraints. That is why there have been developed various frameworks directly aimed at ontology representation, usually based on first order predicate calculus[3] (usually enriched at least by one second order construct — a reification mechanism, which allows the treatment of statements of the language as objects in their own right, thereby making it possible to express statements over these statements). We describe the most significant of them in the following list:

KIF Stands for Knowledge Interchange Format. Primarily intended for cooperation between software agents, as stated in the following definition (according to [20]):

> KIF is a language designed for use in the interchange of knowledge among disparate computer systems (created by different programmers, at different times, in different languages, and so forth). KIF is not intended as a primary language for interaction with human users (though it can be used for this purpose). Different computer systems can interact with their users in whatever forms are most appropriate to their applications (for example Prolog, conceptual

3. Mainly because it is decidable and still quite expressive at the same time.

> graphs, natural language, and so forth). KIF is also not intended as an internal representation for knowledge within computer systems or within closely related sets of computer systems (though the language can be used for this purpose as well). Typically, when a computer system reads a knowledge base in KIF, it converts the data into its own internal form (specialised pointer structures, arrays, etc.). All computation is done using these internal forms. When the computer system needs to communicate with another computer system, it maps its internal data structures into KIF.

Perhaps one the most popular cases of KIF ontology are SUO (Standard Upper Ontology) or SUMO (Suggested Upper Merge Ontology) (see `http://suo.ieee.org/`). These ontologies are designed mainly with respect to e-commerce, educational and natural language understanding application areas.

CycL The ontology representation language used in Cyc project (see `http://www.cyc.com/` or `http://www.opencyc.org/` for a similar open source version). Cyc is aimed at (manual) creation of a system that offers a very large, multi-contextual knowledge base and inference engine. Again, it should enhance software agents' interoperability by constructing a foundation of basic "common sense" knowledge — a semantic substratum of terms, rules, and relations — that will enable a variety of knowledge–intensive products and services. Cyc is intended to provide a "deep" layer of understanding that can be used by other programs to make them more flexible (according to statements at `http://www.cyc.com/`). The structure and application domain of Cyc ontology is thus very similar to the SUO and SUMO ontologies mentioned above.

DAML/DAML+OIL DARPA Agent Markup Language. The OIL acronym stands for Ontology Inference Layer then. These two frameworks together form an ontology representation language then. The language is intended to support semantic annotation of the world wide web based upon XML and RDF (Resource Description Framework) standards. Thus would be the web content accessible not only for human, but even for automated agents of all kinds. The language is a part of World Wide Web Consortium (W3C) standards and contains many constructs for efficient yet expressive representation of knowledge in the form of ontologies, such as reification and formal semantics and

reasoning services provided by description logics. Although we can find ontologies created in DAML+OIL, it is only a predecessor of current World Wide Web Consortium standard for ontology representation — OWL language.

OWL Stands for Web Ontology Language and is defined by W3C. It is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema by providing additional vocabulary along with a formal semantics. OWL has three increasingly-expressive sublanguages (adopted according to document describing basic OWL features at `http://www.w3.org/TR/owl-features/`):

- *OWL Lite* supports those users primarily needing a classification hierarchy and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1. It should be simpler to provide tool support for OWL Lite than its more expressive relatives, and OWL Lite provides a quick migration path for thesauri and other taxonomies. Owl Lite also has a lower formal complexity than OWL DL.
- *OWL DL* supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class). OWL DL is so named due to its correspondence with description logics, a field of research that has studied the logics that form the formal foundation of OWL.
- *OWL Full* is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full.

There are also other ways of representation of knowledge resources that do not benefit from a universal language. This approach is characteristic for proprietary products using their own representation frameworks. On the other hand, a well known WordNet [16] and EuroWordNet [46] project use textual or XML format respectively. Even though the WordNet lexical database is not an ontology in the meaning of a precise definition, it is worth of mentioning here — mostly because it is very closely related to natural language ontologies as mentioned in the next section.

2.2.2 Ontology Classifications

We can distinguish between various kinds of ontologies according to their structure, complexity, scope of content and other criteria. Possible classification categories are overviewed in the few following paragraphs. Note that the classification is not strictly formal and there can be conceptual overlaps in the terms introduced. We mention the classification here rather for technical terminology purposes than to formally discriminate between different ontology types.

Ontologies that define general categories without focusing on a specific branch are called *upper*, *core* or *top* ontologies. We can also have a very abstract ontology that covers even a core one with few most general concepts. Such a level is called *foundational* ontology (see [18] for details). *Domain* ontologies are those that that take into account only a small and well-restricted part of human knowledge. The level of concept instantiation and amount of logical constraints rises when moving from foundational to domain ontologies. This is similar to another ontology classification that distinguishes between *vertical* and *horizontal* ontologies.

Horizontal ontologies are general in nature. These are common ontologies that span multiple domains and provide a mechanism to organise and standardise information content. Vertical ontologies, which also incorporate features from horizontal ontologies, are domain specific. Vertical ontologies not only define data in terms of semantics native to a particular vertical industry, they also contain rules and formal computer languages that can perform certain types of runtime automated reasoning.

2.2.3 The OLE Approach

The OLE ontologies are primarily processed and shared among particular OLE components in a special internal format (see Section 5.3 for details). This resembles the comment on KIF in Section 2.2.1 (agents are storing

knowledge for their own in their efficient "language", but they output and share the knowledge with the environment in agreed and common representation).

We have chosen the OWL DL sublanguage of OWL as a base for external representation of OLE ontologies. This allows us to store knowledge using all essential constructs — classes, subclasses with multiple inheritance, complex properties, reification and so forth. Yet we have a system that is decidable by means of description logics. However, we still need to encode possible uncertainty within OWL DL somehow. We devise special OWL constructs (inspired by [48]) for representing uncertainty that have to be taken into account by uncertain inference engines and can be omitted by classical inference engines at the same time. We present complete description of these constructs in Section 5.3 that is dedicated to specific issues of OLE knowledge representation formats.

OLE is primarily intended to work with complex domain specific (vertical) ontologies. However, we build our ontologies in bottom-up manner — we continuously extract data from particular resources and incorporate them into the complex ontology. An ontology corresponding to a single resource is called *miniontology* and is quite shallow even though it can be (and usually is) very specific. But we consider even other than domain specific ontologies for future. These are mainly a universal natural language ontology, a common sense ontology and an upper ontology for similar domains (acquired and maintained by interoperation between more OLE system instances for various domains).

Chapter 3

Methods of Ontology Acquisition

We have hopefully clarified what is the ontology and what is it good for in the previous chapter. The crucial question is how to build an ontology. Many approaches exist for different ontology utilisations. Relatively many kinds of applications designed for so called "ontology engineering" have also been developed. In the following, the main approaches are shortly described. In the final section, our way of ontology building is introduced and glossed with respect to the previously mentioned approaches.

3.1 Manual Methods and Related Frameworks

One way of creating an ontology (we call it manual development, though the process in general may not be necessarily purely manual) can be described as follows:

1. Let a group of domain experts design the conceptual structure of the ontology more or less manually in some way.

2. Take the results from the domain experts and let a group of computer science specialists formalise the structure to the form ready for the software implementation.

3. Let a group of software engineering experts develop the ontology data representation and possibly some tools for the ontology management and ongoing maintenance as well.

Note that the groups mentioned in the individual points may overlap or even be equal (for example when some computer experts are going to make an ontology comprising the computer science domain).

This way of ontology building was (and to some extent still remains) popular in many ontology engineering frameworks, such as Protégé, WebODE, OntoEdit and so forth. An elaborated overview of such frameworks and/or systems is given in [4] or [8]. We will not describe all the efforts in

this area in detail here. However, some kind of a common overview of the method is given in the paragraphs below.

The discussed frameworks most usually concentrate on development of some comprehensive workbench that can facilitate all phases of the ontology development in an integral manner. The result of the efforts formulated this way is a tool, which is supporting most of the following basic issues[1]:

- the design of the domain conceptualisation
- the formalisation of the conceptualisation
- the implementation of the formalised conceptual structure (in most cases, this part lies in the implementation of some arbitrary taxonomic structure, where the concrete domain concepts shall be filled in)
- concept insertion
- concurrent processing of the ontology by a group of experts in the meaning of the above points
- ontology validation
- merging of ontologies
- ontology browsing
- ontology management (such as updating, versioning, trimming and so on)

The systems developed with respect to the "handmade" approach may cover all of the issues mentioned in the list quite efficiently. However, even when there are many sophisticated techniques, algorithms and presentation methods involved, the main portion of the work (which is considered to be the ontology construction itself) remains on human.

The trickiest part of the ontology development is its structure design and appropriate concept filling in the meaning of "correct" connections of the nodes of the class/concept hierarchy graph. Even when the domain experts' experience is employed within the conceptualisation design, there always looms a possibility of that another group of experts would create a significantly different ontology. The process of manual conceptual design is always arbitrary and in fact, it may not really be a proper representation of the real world subset which happens to be covered. And just the precision

1. According to the statements presented for example in [4], [8], [29] or [42].

of the representation is very often crucial for the efficiency and reasonability of an ontology application.

We need our ontologies to represent the current domains exhaustively and empirically. We do not want to know the layout of the domain conceptual structure as imposed by some more or less personal opinion of an expert group. The first argument, which is underlying such statement was mentioned in the previous paragraph. The second one is that we would need a group of experts for each possible domain of our portals' coverage. This is obviously infeasible, at least when we want to represent knowledge for domains of our concern within some integral framework and also possibly share the knowledge between the respective ontologies. That is why the "handmade" approach was cast aside in the beginning of our research already.

3.2 Overview of Autonomous Techniques

Automatic extraction methods and algorithms are listed in this section. We do not give a complete overview of all possible techniques here, but rather describe the major ones with respect to OLE framework.

3.2.1 Pattern–based Extraction

A pattern based method that can be used for empiric ontology acquisition was firstly sketched by Hearst in [24]. The method consists of automatic pattern-based extraction of particular semantic relations — the hyponymy relation in the case of the referenced work. The hyponymy relation (and its inverse – the hyperonymy)[2] serve well as a basis for an ontology structure in the meaning of the natural hierarchy of conceptual classes. However, the notion of the automatic extraction of hyperohyponymical constructs from the resource data can be adopted for any other existing relations (such as synonymy, antonymy, meronymy – the part to whole relation, holonymy – the whole to part relation and so forth).

The hyperohyponymy constructs a well defined and relatively easily acquirable concept hierarchy in the meaning of superconcept/subconcept relations. The relation directly corresponds to the aggregation of specific concepts into classes (and also of some classes into superclasses), which is one of the backbones of most of the ontology encoding formalisms. See Chapter 4 for details on the algorithm and it's implementation in OLE.

2. Less formally, both of these relations are often introduced as the *is-a* relation by some authors.

3.2.2 Concept Sets Acquisition

In addition to the technique of the pattern-driven extraction of semantic relations, other methods based on token co-occurrence in the resource data can be employed in order to gather sets of concepts belonging to the same class. Various modifications of these generic techniques are listed in [34]. The existing lexical databases may be used for discovery of concept trees — for example in the OntoLearn project [18], the statistical methods based on document frequency measure are equipped in order to extract terminology from the source data. Then the WordNet database is queried in several stages of the semantic interpretation and specific relation discovery. WordNet is used as auxiliary resource also in the proposed process of automated meta-data hierarchy creation (see [25]), which is partially related to the automatic ontology acquisition as well.

Another important way of how to obtain concept sets from data utilises one of the unsupervised machine learning techniques — clustering. We find the clustering most appropriate for our empiric approach to ontology construction, because it allows us to directly extract even a taxonomy of the concepts from a text using so called hierarchical clustering. See Chapter 4 for implementation details.

3.2.3 FCA and FFCA Approach

FFCA stands for Fuzzy Formal Concept Analysis (as FCA stands for Formal Concept Analysis). Quan et al. propose the method in [43] as yet another technique of automated ontology generation, based on concept clustering. The technique has the notion of uncertainty implicitly embraced already at the initial level of information extraction.

The FFCA method is proposed to be utilised in automatic creation of scholarly Semantic Web[3]. It generates so called *fuzzy formal context* from a citation database. From the *fuzzy formal context*, the *fuzzy formal concept lattice* is constructed. When the lattice is gained, the concepts on it are clustered and form the *concept hierarchy* then. The hierarchy is directly transformed into an ontology, respecting these rules:

- Each concept is represented as an ontology class[4].

3. The content of the following paragraphs of this section is adopted according to [43]. The proper definitions, conceptual clustering algorithm, various examples and valuable relevant references can be found there as well.

4. Note that the authors use slightly different terminology than the one being used in this paper — the *concept* and the *object* terms are analogous to the *class* and *instance* terms re-

- The subconcept and superconcept relations from the concept hierarchy are preserved between the classes respectively.
- Concept attributes become properties of the corresponding class.
- Each object in a concept is represented as an instance of the corresponding class.
- The value of an instance property is the membership value (in the meaning of the fuzzy formal context definition, see [43]) of the corresponding object's attribute.

Currently, we postpone the uncertainty to the phase of ontology merging. But the approach proposed by Quan et al. for domain meta-information processing (the citation information) seems to be very promising even for the domain content representation efforts (our case). The method, possibly adjusted and adapted to our needs, can be implemented as another information extraction plug-in for the OLITE system.

3.3 Discussion of Methods Used in OLE

The acquisition of *is-a* relations is essential for OLE. If we can discover the relations in the resource data, then we can create the respective taxonomy of the resulting ontology directly in the bottom–up manner (from ontologies with very simple structure to more complex ones in a continual iterative process). And when such a process is fully automated, we can acquire and update our ontologies dynamically from the submitted real world data. The ontologies reflecting the particular resources are merged into the current domain ontology by another specialised software agent. The result is no way arbitrary and subject–dependent. In reverse, it is purely empiric and as up to date as are the resources used when considering the current state of the domain. This approach complies very well to the remarks in [37]. Although the topic covered by this reference is not directly related to the OLE project, its proposals are to some extent universal, suggesting to build the Semantic Web as simple as possible. Knowledge representations should not be artificially structured in an ambitious effort to devour the whole world in an ontology. According to [37], the mappings between the "local" knowledge repositories should not be some standardised rigid translation rules,

spectively. We use the *concept* term for any relevant token extracted from the resources. The difference between classes and instances is not stressed until the miniontology generation and merging phases take their places (see Section 4).

but rather dynamic principles that pay attention to the current situation. More or less, this proposal can be stretched out to the automatic merging of simple miniontologies into the richer current domain ontology (using a simple technique described in Section 4.3 or even the principles of uncertain information representation, as sketched in Chapter 5).

Thus we propose a flexible ontology building approach, which is based on automated concept extraction from mainly unannotated plain text resource data. No manual conceptual engineering is performed, the generated ontologies are simply mirroring the respective domain according to dynamically submitted relevant resources. However, manual adjustments are always possible and they can be made by an external tool (such as Protégé).

Virtually any method of automated knowledge acquisition introduced in the previous two subsections can be employed as an independent part of the OLITE module. Presently the following methods have been examined — pattern–based extraction of semantic relations and hierarchical clustering improved by automatic class annotation.

A so called miniontology is created for each input resource, comprising the concepts extracted from the resource. The pattern based method suffers from data sparsity (as the automatic methods often do). We may have not a sufficient number of relation patterns in data and thus the overall coverage of the system would be quite low. That is why we incorporate a more extensive method based on hierarchical clustering of words for building the basic domain ontology (see Section 4.2). This method processes all nouns in a resource and so it is less sensitive to the data sparsity problem. Then, the miniontologies produced by pattern–based extraction are integrated into the current domain ontology on the fly. Currently a very simple method is used for the complex task of ontology integration (see Section 4.3). The resulting ontology can be further improved and enlarged by another application of clustering when the amount of new resources is reasonably high. However, it is not practical to perform the clustering very often because it is very resource–demanding.

As sketched in Section 5, the process of proper automatic ontology merging and alignment would embody the application of uncertain knowledge representation methods (even if the currently produced miniontologies are encoded in the classic "certain" way). The concrete structure of OLE tools is developed in Chapter 6.

Chapter 4

Autonomous Acquisition of Ontologies in OLE

In the following two sections we describe two extraction algorithms that have been incorporated in OLE[1]. In the third and last section of the chapter there is depicted a simple technique used for the merging of extracted ontologies. However, more sophisticated techniques are needed when we want to perform really ”intelligent” ontology integration. This topic and respective framework is discussed further in consequent Chapter 5.

4.1 Extracting Taxonomy Chunks Using Patterns

As stated above, the pattern–based method is based on the work presented by Hearst in [24]. Supposing we know patterns characteristic for a semantic relation (namely the *is-a* relation), we can extract the reference words of concepts involved in the relation from a text.

4.1.1 Preprocessing of the Text

First we need to process the resource in order to gain plain text of relevant sentences with respective representation of their syntactic structure (at least a shallow one). This preliminary phase has to be as fast as possible so we use simple but computationally efficient techniques. The possible lack of accuracy is balanced by the large amount of processed data. The preprocessing is performed in the following steps:

1. *Extraction of plain text* — The OLE preprocessing module is designed to work with plain text, that is why the text must be extracted from a resource if needed. There are many publicly available tools able to do this (such as UNIX `html2text` utility for HTML documents).

1. Note that because the automatic acquisition of conceptually valid ontologies with complex relational structure is often considered as a very difficult or even infeasible task, we have concentrated mainly on taxonomy structure extraction so far.

2. *Splitting of the text into sentences* — Each pattern occurs within a sentence. Therefore it is desirable to split the text into sentence tokens. We do this using regular expression matching. This can be less effective than for example machine learning techniques, but it is very fast and yet quite precise for our purposes (the ratio of correctly split sentences varies between 85 and 95 percents depending on the kind of sources).

3. *Elimination of irrelevant sentences* — We do not have to process the sentences with no pattern present, so we eliminate them in this step, using regular expression matching again. However, the patterns are general so we do not perform straight matching of text tokens. We rather examine matches of sentences with compiled pattern objects that wrap the basic regular expression operations (see Section 4.1.1 for details). Thus we discover sentences that *possibly* contain a relation and eliminate those that certainly do not contain one. This minimises the data load for further stages that are slightly more demanding then simple one–pass regular expression matching.

4. *Tokenization and tagging of the relevant sentences* — Before finding out a syntactic structure of a sentence, it has to be divided into word tokens with respective morphology (part of speech or POS) tags. The tokenization is regular expression based again. For POS tagging we incorporate Brill transformational tagger (see [7] for details) combined with statistical and regular expression taggers. We do not use stemming, for we could lose important information regarding specific individual concepts.

5. *Chunk parsing of the sentences* — Eventually, we parse the sentences with custom chunk parser that identifies prepositional, noun, verb and so called core phrases in the input sentences. The core phrases are those that identify and form the core of a pattern. When we have identified also the phrases that are surrounding the core phrase, we can directly extract the respective semantic relation between the concepts involved.

Language Universality

The only phase which is language dependent in ontology acquisition is the preprocessing of data. If we can preprocess a text in the steps described above and know the proper patterns, we can also extract the respective re-

lations and concepts. At the Figure 4.1 is schema of the tools ensuring language independence of OLE under some conditions. These conditions are mainly existence of respective corpora for the given language.

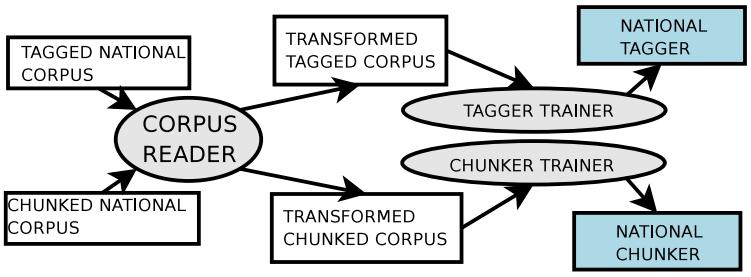


Figure 4.1: Language universality support

The components depicted at the figure are described as follows:

Corpus reader interface implements sets of rules that transform an existing corpus of particular language into the format comprehensible by OLE tools.

Tagger trainer utilises a tagged corpus in order to create a respective POS tagger; stochastic and Brill transformation based taggers are trained to be combined with manually designed regular expression and default taggers that are characteristic for the current language.

Chunker trainer utilises a treebank-like corpus in order to learn how to chunk the input tagged sentences and identify particular phrases in a text. However, it is very useful to combine the rules learned this way with few additional (manually designed) rules that comprise at least core phrases for pattern identification.

Creation of Pattern Objects

The pattern–driven extraction process accepts patterns in a special form. The designed universal pattern–specification format allows new patterns to be easily added in the future.

The patterns are loaded and compiled from a separate PES XML file. PES stands for *Pattern Extended Specifications*. The syntax of expressions occurring in a pattern is similar to extended regular expressions. Moreover, the pattern structure is encoded via structure of respective XML elements.

The PES file has several levels that represent a kind of pattern, an identifier of a pattern and the pattern structure itself. Specific constructs are used to encode positional attributes and meaning of various parts of a pattern as seen in the example below. Respective pattern objects are created after compilation of the PES file. These objects are used for preprocessing and for core phrase identification in the chunk–parsed text when extracting the relations.

The *is-a* pattern (adopted from [15]) in the form of:

```
NP1 {‘‘,’’} ‘‘such as’’ NPList2
```

is transformed into the following PES element within a PES file:

```
<?xml version="1.0"?>
<xml>
  ...
  <rel id="is-a">
    ...
    <patt id="3">
      <b pos="1" expr=".+"/>
      <core pos="2" expr=" such as "/>
      <a pos="3" expr=".+"/>
    </patt>
    ...
  </rel>
  ...
</xml>
```

The `rel` element presents a container for a respective relation (identified by the element's only attribute `id`) patterns. The `patt` element encloses a pattern identified by the only `id` attribute again. There can be three kinds of elements as children of a `patt` element:

`a` We operate only with binary relations in OLE ontologies (any relation of higher arity can be expressed using binary relations when needed). The `a` element in PES pattern represents expression(s) that refer to concepts acting as a first element of the considered relation.

`b` Similarly to the previous point, `b` represents expression(s) that refer to concepts acting as a second element of the considered relation.

`core` This element represents expressions of core words in a pattern.

Each of the elements is provided with `pos` and `expr` attributes encoding the element position within a pattern and corresponding regular expres-

sion. This is all information needed to compile the universal pattern objects for further processing of data.

4.1.2 Extraction from Chunked Text

Concept extraction utilises the abstract regular expression matching again, but it works on the chunked sentences and the compiled PES patterns. The abstract matching means that the objects are not compared as standard strings. They carry information on what they are representing (a chunked sentence or a PES pattern) and what kind of operations should be applied. We concentrate on creation of taxonomies. The taxonomic relations hold especially for nouns. Although we could imagine hyperohyponymic strings for other parts of speech, extraction of the noun hierarchy is the only reasonable way of how to obtain an ontology skeleton. The taxonomy pieces are extracted directly according to the respective patterns.

In the Table 4.1 we give an example of few hand–crafted patterns and their meaning (partly adopted from [15] and [24]). The patterns are introduced in an intuitive regular expression based form, where NP stands for a simple noun phrase and NPList stands for simple noun phrases conjugated by a comma or a coordinating conjunction. By a simple noun phrase we mean a noun or noun compound, supplemented by possible modifiers. In case of the extraction of taxonomies we need not to be concerned about those supplements.

Selected *is-a* pattern	It's interpretation
NP such as (NPList | NP)	The concept(s) referred by the nouns on the right are hyponyms of the concept(s) on the left
NP including (NPList | NP)	The concept(s) referred by the nouns on the right are hyponyms of the concept(s) on the left
NP (is | was) an? NP	The concept referred by the nouns on the left is hyponym of the concept on the right
NP like (NPList | NP)	The concept(s) referred by the nouns on the right are hyponyms of the concept(s)on the left

Table 4.1: An example of patterns for *is-a* relation

We focus at noun phrases appearing in the vicinity of core phrases. The algorithm implemented in OLE is non–formally described as follows:

input: chunk–parsed sentence
output: list of *is-a* relations

1. for each matching pattern within a sentence, create a triple(s) (l, c, r), where l, c, r are left context, core phrase and right context respectively;

2. for each triple, get the complex noun phrases that are nearest to the core phrase;

3. assign the hyperonymy/hyponymy mark for these phrases according to the respective matching PES pattern object;

4. extract the hyperonyms/hyponyms from each noun phrase NP in the following manner:

   4.1 if NP is a simple noun phrase, extract the head noun from the phrase and return it as single hyperonym/hyponym;

   4.2 if NP is a conjugated noun phrase consisting of several simple noun phrases, extract the nouns or noun compounds form each one of them and return them in hyperonym/hyponym list;

5. join all the hyperonyms/hyponyms gained in step 4. in respective *is-a* relation instances;

6. return the relation instances;

The extracted information is stored in a universal internal format. An output miniontology file can be produced by applying respective translation rules. See Chapter 5 and Section 5.4 for further information on the internal representation and dumping of ontologies. However, the miniontology is primarily passed to the integration phase. Basic algorithm for ontology merging is given in Section 4.3.

4.2 Extracting Concept Sets Using Hierarchical Clustering

We use the hierarchic clustering and consequent WordNet–based annotation of resulting classes in order to acquire sets of individuals and their classes' hierarchy, which naturally corresponds to an ontology taxonomy.

The approach is somehow similar to work presented by Lin and Pantel in [35] in the context of automatic extraction of senses from a text, although we rather discover relations between groups of words than their senses. Because of extensive exploitation of the provided data (all present nouns are taken into account), we use this method to create the initial domain ontology.

4.2.1 General Algorithm

According to the survey in [45], the algorithms for word clustering are classified into two types. One type is based on construction of an initial set of classes and further shuffling of words among these classes until the clusters are stabilised. The other type utilises iterative merging of classes that initially contain only one word. Both types are driven by an objective function, in most cases by perplexity or average mutual information. Efficient clustering can also be made using more general distance measures in a vector space that represents the input data. These methods are directly available for OLE due to incorporation of NLTK [1] natural language processing framework. Namely the cosine distance of normalised vectors is used as a measure for vector similarity in OLE.

The latter type of word clustering is most appropriate for extraction of taxonomies, because we can easily convert the history of the merging process to a tree–structured representation of the input data. The first type cannot be directly used for obtaining a taxonomy and is very dependent on the initial set of classes. Finding a good initial set is itself a very difficult problem, so we adopt the second type of clustering for OLE. The main benefit of clustering techniques in the scope of OLE is that they are unsupervised and run fully automatically without a need of human assistance.

4.2.2 Preprocessing, Dictionary Creation and Feature Assignment

The preprocessing need not be as complex as for the extraction of semantic relations — usual stop list application, text tokenization into words and POS tagging is fully sufficient (we do not use stemming again). Since we want our data to be viewed as a vector space before we can cluster them, we have to encode the vectors that correspond to words we are going to process. By these vectors we represent the context words relevant for clustering.

The clustering demands a relatively large dataset. Having this dataset, we create a domain dictionary of all the words present in the set. Each word

has a unique ID (natural number) assigned after adding it into the dictionary (it can be for example the order in the sequence of word additions). These IDs act as elements of vectors that characterise the words to be clustered then.

When we have annotated the text with respective IDs, we extract the nouns with their respective symmetric contexts from left and from right. The size of the context is defined empirically — size of ten words for the whole context was found to be sufficient for basic evaluation. The context words form a feature vector we assign to the extracted noun.

4.2.3 Hierarchical Clustering and Class Annotation

By the hierarchical clustering we obtain classes of words, classes of these classes of words and so forth until we have a root of the respective tree (called a dendrogram in the field of machine learning). The history of merging steps allow us to reconstruct a noun taxonomy from the dendrogram, but the classes at particular levels are anonymous. If we want to translate them into an ontology, we have to annotate them by names that conform to hyperonymy relation in the context of words comprised by the respective class. The bottom level (dendrogram leafs) represent individuals in the ontology then, as the upper levels represent classes. We do this more or less empirically, using the WordNet lexical database. Sometimes we use the CAANNO (Clustering and Autonomous ANNOtation) acronym for the technique in the following text. The novel algorithm for CAANNO acquisition of a named–classes' hierarchy is described as follows:

input: vectors corresponding to nouns associated with their contexts
output: internal representation of an ontology taxonomy

1. induce a dendrogram from the preprocessed input data;
2. for each level in the dendrogram tree (going from the bottom to the root), perform the following:
   - 2.1 to all words in each class at the current level, assign a set of all hyperohyponymic strings in WordNet they are involved in;
   - 2.2 compute the most frequent hyperohyponymic string among each class;
   - 2.3 from the nearest hyperonym in this string (that is not present as an individual in a class), make an annotation for this class

(take an arbitrary word from the respective synset and update the domain dictionary if needed);

3. translate the annotated tree into an ontology representation;

The step 2.2 in the above algorithm forms a computational bottleneck for autonomous class annotation — we should mutually compare all hyperohyponymic string with respect to all senses the involved words can have. This causes very undesirable combinatorial explosion.

That is why we use a heuristics to overcome this problem. It is very simple — when computing the string frequencies among a class, we process only strings corresponding to all senses of all words in a class in one pass. The addition of each string that is corresponding to a word w is weighted by the $\frac{S_{avg}}{S_w}$ modifier, where S_{avg} is average number of strings per one word in the given class and S_w is number of strings for the word w.

Such weighting corresponds to an intuitive idea that the words with more hyperohyponymic strings assigned should not overwhelm the words with less strings in the frequency computation. This heuristics solves the combinatorial explosion problem and yet it does not reasonably harm the annotation usefulness, as seen in Chapter 7.

4.3 Domain Ontology Creation and Iterative Enrichment

Supposing we have created a basic domain ontology, we need to process new resources that are coming to OLE. Re–clustering after each deployment of a new resource is computationally demanding. But we can use the efficient pattern–based extraction method combined with classification by current cluster structure. Taxonomy chunks extracted by the pattern–based method enrich the domain ontology structure in the meaning of spreading of leaf groups of words and possible assignment of new and previously unknown relations between classes. The algorithm of addition of a new resource is given below:

input: current domain ontology, new resource
output: enriched domain ontology

1. process the resource by a pattern–based module and create the respective miniontology;

2. extract the context–vectors for each term in the miniontology from the resource;

3. for each instance $isa(c_1, c_2)$ of *is-a* relation between concepts c_1 and c_2 from the miniontology, compute domain ontology concepts $\hat{c_1}$ and $\hat{c_2}$ respectively as the concepts with the lowest cosine distance from c_1 and c_2;

4. compute c as the nearest common hyperonym of $\hat{c_1}$ and $\hat{c_2}$ in the domain ontology;

5. set the new relations $isa(c_2, c)$ and $isa(c_1, c_2)$ in the domain ontology, possibly marking c_1 as an individual and c_2 as a class (if their states were different in the former domain ontology);

6. return the updated ontology;

The step 5. in the above algorithm contains operations of marking a concept as an individual or as a class. This is caused by the fact that obviously only concepts that have no subclasses can be individuals in the meaning of statements given in Section 2.1.3 and Section 2.2. Thus we empirically set the concepts without subclasses in the updated ontology as individuals. On the other hand, the concepts that have had no subclasses before and have ones after merging with miniontology must be marked as classes instead of individuals in this perspective.

The presented method is very rudimentary and relies on the correctness of relations present in both miniontology and domain ontology. However, we do not know whether the relations really are correct or not. A mechanism for analysis of correctness measure assignment and utilisation of this measure in the process of proper empiric ontology merging is needed. Uncertain reasoning would be very efficient for these tasks. In the following chapter we introduce a framework for uncertainty representation in OLE ontologies. Such framework serves as a basis for respective reasoning tools that are needed for proper autonomous ontology merging.

Chapter 5

Remarks on Uncertainty Representation

The significance of uncertainty representation has become obvious in the Semantic Web community recently. A new framework for uncertain information processing is presented in this chapter. Formal systems that underlie the uncertainty representation are briefly introduced. We discuss a universal internal format of uncertain conceptual structures in OLE then. We also present details on translation from such format to the common OWL language (with extensions regarding the uncertainty).

5.1 Motivations

Ontology integration phase is the moment when the need of uncertainty representation arises. Even if we could obtain precise conceptual constructions from single resources (e. g. *birds fly*), we will experience infeasible consistency difficulties when trying to assign precise relations between the concepts in broader scope of the whole domain (as illustrated by the popular example: the fact *birds fly* collides with the statements *penguins are birds; penguins do not fly*). Besides the inconsistency handling, there are also important cognitive motivations of the utilisation of uncertainty in our empiric ontologies that led us to the proposal of a novel framework for representing uncertain knowledge. It is called $\mathcal{ANUIC}$ ($\mathcal{A}$daptive $\mathcal{N}$et of $\mathcal{U}$niversally $\mathcal{I}$nterrelated $\mathcal{C}$oncepts).

The knowledge repositories built by OLE tools must reflect the state of the respective domain empirically according to information contained in the provided resources. Such kind of knowledge is as much objective as possible, because it is not influenced by arbitrary considerations about the domain's conceptual structure, but determined by *the structure itself*.

Remedy to Emerging Inconsistencies

However, the automated empiric approach has an obvious drawback – the threat of inconsistency. As we do not generally have an infallible "oracle"

to tell us how to precisely join or map newly extracted concepts to the ones that are already stored in our ontology, crisp relations between concepts are virtually impossible. We must deal with the inconsistencies somehow.

There are two general kinds of possible inconsistencies in an ontology (virtually any relational inconsistency can be modelled using these):

- *subsumption* inconsistency: given concepts C, D and E, the $C \subseteq D$ and $C \subseteq E$ statements may collide when we represent for example crisp *part-of* relation by the $\subseteq$ symbol (e. g.: Turkey is both part of Europe and Asia)
- *equivalence* inconsistency: given concepts C, D and E, the $C \equiv D$, $C \subset E$ and $D \equiv E$ statements are in conflict (for example when we find out in a text that *'science'*, *'knowledge'* and *' erudition'* are synonyms and at the same time we induce that *'knowledge'* is a super-concept of *'erudition'*)

Such collisions are hard to be modelled in classic crisp ontology representation frameworks (see [27] or [48]). Implementation of the uncertainty into our knowledge representation is a solution for dealing with conflicts in the continuously updated ontology.

Mental Models Reflection

The second motivation lies in inspiration by the conceptual models that are characteristic for human mind. This topic is closely related to the very definition of *concept* and *meaning*. As stated for example in [26] or [11], people definitely do not represent the meaning of concepts as static crisp structures. The meanings are rather constructed as vague sets of dynamically overlapping referential associations [26], or so called "meaning potentials" with particular instantiation dependent on the context of concept-referring word or sequence of words [3].

We need to develop a little bit more precise definition of a concept than the "technical" one that was introduced in Section 2.1.3. Other formulations related to this topic are presented in section 5.3. By an $\mathcal{ANUIC}$ concept we mean a representation of an entity existing in real world and/or utterable in human language. A concept is determined by its relations to another concepts in the universe then. Such "relational" definition of a concept is partly inspired by poststructuralistic philosophy (see for example [12]). Reference of a concept is then realised by instances of its relational connections. By these instances we mean especially concrete uncertainty measures assigned to each relation a concept is involved into (see Section 5.3 for details).

Thus we can naturally represent the dynamic conceptual overlap in the meaning of [26], because the assigned relations' measures are continuously updated within the process of a new knowledge incorporation. And by introducing a special relation of *association* we can represent the notion of meaning potentials according to [3]. Using this relation we can associate a concept with a representation of co-occurring concepts and impose another useful restriction on the meaning construction (helpful for example when resolving word-sense ambiguities).

5.2 Theoretical Background

In this section we very briefly recall the main approaches to representation of uncertainty. The uncertain information representation frameworks are determined by three significant fields of contemporary mathematics:

1. extending the theory of measure into a more general theory of monotonous measures with respect to the classical measures of information
2. applications of (conditional) probability theory
3. extending the classical set theory into a more general fuzzy set theory

Extended Probabilistic Frameworks

Various uncertain extensions of the information measure theory are mentioned by Klir in [28]. However, in the computer science field there are other probabilistic theories generally accepted, mainly in the scope of:

- Bayesian networks
- non-monotonic reasoning and respective probabilistic (or possibilistic) extensions of "classical" (mainly propositional, first order or description) logics

The former approach exploits the idea of a specific graphical model (Bayesian network) of a variables (nodes) connected by respective conditional probabilities (edges) in a DAG. A good overview of this framework and related algorithms offers Xiang in [47]. For example Peng [48] or Holi [27] offer an application of Bayesian networks for uncertain ontologies.

The latter approach supplements the classical logics' paradigms with uncertain (e. g. conditional) consequence relations. The reasoning within these frameworks is inspired by both of extended theory of information

and (conditional) probability theory. Good overview of these approaches is given in [23]. In the scope of ontology reasoning there are probabilistic extensions of description logics that was designed in order to tackle reasoning tasks among crisp ontologies – for example Lukasiewicz's P-$\mathcal{SHOQ}(\mathrm{D})$ logic [21].

All these more or less probabilistic approaches are no doubt significant for uncertainty representation. However, we dissociate from them in our work for one main reason. As we want our ontologies to be built automatically in an empiric manner, it would be very hard to find out appropriate (conditional) probability assignments (especially in cases when the input data are sparse in some subdomains of our interest) without any background knowledge (axioms and/or inference rules) at our hand. That is why we prefer using the fuzzy sets and fuzzy logic formalisms to motivate our uncertain knowledge representation proposal.

Fuzzy Sets

Fuzzy sets were introduced by Zadeh [49]. Each fuzzy set is uniquely defined by its *membership function*, which assigns a certain degree of respective set's membership to each element in the considered universal set X. The membership usually ranges in real numbers interval of $[0, 1]$. For a set A we denote the membership function as A here, so $A(x)$ means the membership degree of the element x w. r. t. A. These fuzzy sets are called standard. For each standard fuzzy set there can be defined a standard crisp set $A^{\alpha} = \{x \in X | A(x) \geq \alpha\}$, which is called α-*cut* of the set A. A fuzzy relation R on $X \times X$ is then defined as a mapping $R : \ X \times X \ \rightarrow \ [0, 1]$. Notions of reflexivity, symmetry, transitivity etc. similar to those of classical relations can be adopted even for fuzzy relations. This is very useful for example for reasoning tasks (see [19]) based on set operations. However, this intriguing topic will be discussed elaborately in another dedicated paper.

Many variations branching from the original Zadeh's idea have been developed until now. Some of them (besides the original fuzzy sets) are quite significant with respect to our research topic. The **fuzzy rough sets** and **rough fuzzy sets** (as introduced by Dubois and Prade in [13]) are based on approximations of (fuzzy) sets using a fuzzy similarity relation or crisp equivalence classes. Using these theories we can structure our conceptual universe as such approximation space in various perspectives (according to the relation used). **Intuitionistic fuzzy sets** (see [9]) based on combination of membership and non-membership degree can be used when dealing with negative knowledge in our ontologies.

5.3 Proposal of $\mathcal{ANUIC}$ Format

$\mathcal{ANUIC}$ ($\mathcal{A}$daptive $\mathcal{N}$et of $\mathcal{U}$niversally $\mathcal{I}$nterrelated $\mathcal{C}$oncepts) forms a backbone of the uncertainty representation in OLE. The formal definition of $\mathcal{ANUIC}$ and issues regarding possible problems, modifications of basic empiric approach as well as reasoning perspectives are mentioned in this section.

5.3.1 Formal Definition

The concepts are stored in a special fuzzy network structure. The network is an oriented multigraph $G = (V, E)$, where V is a set of stored concepts and E is a set of ordered tuples (u, v), where $u \in V, v \in V$ (more edges can appear between two nodes). The edges are induced by imprecise concept relations. Multiple edges are allowed as there can exist multiple relations between concepts. A node is a tuple in the form of (c, R, A), where:

- c is a **core** word of the concept. It serves as a master reference index and is computed as the most frequently occurring word in the scope of the hyperohyponymy relation instance with the highest associated μ-measure (see below what the μ-measure is). The hyperohyponymy relation was chosen because it is commonly considered as a basic relation when forming a knowledge basis.

- R is a **relational** set of tuples in the form of $(r, c_r, \mu(r))$, where $r \in N$ is an identifier of a relation from a given set N (its members can be usual lexico-semantic relations, such as hyperohyponymy, holonymy, meronymy, or domain-specific relations like *used_for*, *appears_in*, *method_of* and so forth). The $c_r \in V$ is again a concept, which is related with the current one by r, and $\mu(r) \in [0, 1]$ is the μ-measure assigned to this observation.

- A is an **associative** centroid vector of words appearing in the context of the core word (thus being a representation of the most common context of a concept). However, the word vector is too specific, so it is more reasonable to encode the context in the form of general concepts covering the most frequent word occurring in the context. These upper concepts are assigned from the current ontology. The notion of an associative set supports the meaning potentials remark from Section 5.1 as well as the feature assignment mentioned in Section 4.2.

5.3.2 Conviction Function

For computation of the μ-measure (that is, the membership/appropriateness function value) $\mu(r)$ for a relation r that is corresponding to a (c_1, c_2) edge we devise the following heuristic "conviction" formula (derived from the standard sigmoid function):

$$\mu(r) = \frac{1}{1 + e^{-s(f_r - \alpha)}}$$

where $f_r = \frac{f(r(c1,c2))}{\sum_{c \in V} f(r(c1,c))}$ is the relative frequency of relation observations in input data, s is a parameter regulating the "steepness" of the function and α influences the placement of the inflexion point. The domain of the function is real interval $(0, 1\rangle$ (but only rational numbers obviously appear as an input). The range is real interval $(0, 1)$.

Proper adjustment of the parameters defines the reflection of the impact of frequency on the fuzzy appropriateness μ-measure of the observed relation. Thus we can regulate for example the "conservativeness" of the system (in the meaning of the influence of major or minor observations to the overall conviction). The function is continuous and thus can be implemented in a very straightforward way. However, it can easily imitate discontinuous jumps in the shape of the curve, which is also very useful. Examples showing shapings of the conviction function are displayed in Figure 5.1. In Table 5.1 there are given the parameter values corresponding to the respective shapes of the conviction function. As we can see from

Plot number	Curve name	s	α
1	f(x)	10	0.5
1	g(x)	20	0.5
1	h(x)	20	0.75
1	i(x)	10	0.2
2	j(x)	10	0.75
2	k(x)	2	0.5
2	l(x)	5000	0.97
2	m(x)	6	0.0

Table 5.1: The conviction function parameter settings

these examples, the proposed conviction function allows us to simulate naturally the relative influence the observation frequency has on the relevancy

of the observed relation instance. To be more specific, consider the following overview:

- On the plot with label 1 there are given only slightly deformed curves that are quite similar to the standard sigmoid shape. The functions with α set to 0.5 and sufficiently high s reflect a symmetric impact of the frequency on the μ-measure, raising from 0 to 1.
- The shapes presented on the plot 2 show us the flexibility of the proposed conviction function more illustratively:
  - Shape labelled as $m(x)$ presents quite "hesitating" function that assigns quite high μ-measures (greater than 0.5) even to small frequencies, thus making the system partially believe in almost every evidence, yet preferring higher frequencies significantly.
  - We can also acquire an almost "linear" curve shape (the $k(x)$ label), however it is more convenient to take directly the frequency as the μ-measure if we want a reasonable linear formula.
  - The $j(x)$ function presents a shape assigning relatively low values (in the meaning that they are quite far from 1) even for frequency near or equal to 1. It reflects an "opinion" of the system that even a provisionally sure fact can never be absolutely valid if we consider future observations.
  - The shape given by $l(x)$ presents a very "conservative" settings – only very high frequency will get a μ-measure significantly higher than 0, observations with minor frequencies are ignored. The α parameter presents a threshold of these ignored frequencies here.

5.3.3 Notes on the Interpretation of μ-measures

In the following few paragraphs we present basic ideas related to utilisations of the notions described in the previous section.

1. **Learning and Propagation of the Conviction Function Parameters:** Given a reasonable and comprehensive portion of annotated concept data from an ontology domain, we can learn specific settings of conviction function parameter for particular concepts (besides of selected parameters valid for the rest of an ontology). Thus we can reflect for instance whether a concept tends to have more instances of a relation at a time – the conviction function should assign almost same

high values to almost equal (but relatively low) frequencies. The parameter settings can then spread over transitive strings in ontology within reasoning operations performed on $\mathcal{ANUIC}$ format. However, the annotation of data (and/or perhaps even implementation of unsupervised learning methods) as well as concrete implementation are still mostly subjects of future research.

2. **Data View Perspectives:**
The μ-measures of relations in ontology allow us to impose various perspectives upon the stored data. The primary perspectives are *set-oriented* perspective (e. g. fuzzy set constructed by the μ-measures of subconcepts related to a concept or crisp approximations of the ontology structure given by some specific α-cuts) and *relation-oriented* perspective (e. g. the fuzzy synonymy relation). These perspectives allow us to develop reasoning procedures using the results of the existing theory of uncertain fuzzy inference (see for example [5], [19], [14] or [23]).

3. **Coping with Sparse Input Data:**
The network constructed this way squares with the ideas presented in Section 5.1 and conforms with the very intuitive notion of how people natively represent concepts in their minds. The dynamics of the system rests on continuous updating of all the μ-measures from the observed data. However, the real world data are not homogeneous in the frequency distribution of particular concepts. For some rarely occurring but important words the empirically measure could easily be unsuitable for further utilisation of such knowledge. Therefore additional "referees" must be incorporated especially for terms with low frequency (and even for the other ones). Existing lexical databases and electronic thesauri are good for correcting the possibly invalid uncertain measures gained by empiric evaluation of sparse data. In order to combine more resources of such external judgement, usage of WordNet [16] lexical database with Bonito2 word sketches [38] and Roget's electronic thesauri services [33] is appropriate.

4. **Conscious and Unconscious Operations:**
In an analogy with human mind, two kinds of operations within the $\mathcal{ANUIC}$ knowledge base are possible. We call them *conscious* and *unconscious* operations. The former are triggered by external incentives – mainly observations from the input data, user queries, or administrator commands (for example dumping the knowledge base in order to

examine concept shifts over time later or learning the conviction function parameters). The *unconscious* operations are run by the knowledge base itself and are of the same importance as the conscious ones. These operations are mainly reasoning tasks like merging of concepts that have a reasonable high measure of synonymy or an inverse operation of splitting concepts. They should be run when there are no extensive computational demands on the knowledge base. Proper implementation of such operations helps to improve the consistency and manage the redundancies in the stored data.

5.3.4 An Example of Ontology Fragment Creation

We offer a very simple example of uncertain ontology integration within $\mathcal{ANUIC}$ below. In the Figure 5.2 there is given a sample text as an input for miniontology extraction. The words referring to respective concepts in our fragment are marked with an index that represents them in the Table 5.2.

The Table 5.2 shows relevant *is-a* taxonomic relations (noted as r) before and after the merging of the miniontology with a domain ontology[1]. Both halfs of the table present a matrix with indices from concept set and values from the $\mu(r)$ range (for states before and after the integration). Semantics of a matrix element e_{c_1,c_2} is: "the concept c_1 (the row index) *is a* concept c_2 (the column index) with conviction given by the appropriateness measure e_{c_1,c_2}".

$\mu(r)$	c_1	c_2	c_3	c_4	c_1	c_2	c_3	c_4
c_1	1.0	0.976	0.908	0.0	1.0	0.953	0.881	0.296
c_2	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0
c_3	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0
c_4	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0

Table 5.2: The *is-a* relation μ-measure values before and after integration

5.4 From $\mathcal{ANUIC}$ to (F)OWL

The technical implementation of $\mathcal{ANUIC}$ is based on multiple–level hash arrays with these levels:

1. The μ-measure function with parameters $s = 10$ and $\alpha = 0.2$ is used. The initial relative frequencies for observations $r(c_1, c_2)$, $r(c_1, c_4)$, $r(c_1, c_3)$ are $\frac{4}{7}$, 0, $\frac{3}{7}$ respectively.

- relation identifiers form indices at the top–most level (for example *is-a* ID);
- identifiers of concepts appearing as a first part of the given relation form indices at the mid–level (the concept IDs are assigned according to the domain dictionary, see Section 4.2.2 for details);
- identifiers of concepts appearing as a second part of given relation form the lower–level indices.

Thus we encode the relation instances, with respective instance μ-measures as values at the bottom level of the multiple hash structure. Objects that represent the concepts themselves are stored in another hash structure indexed by their IDs.

When dumping an ontology, we process the hash from the top–most level and assign specific OWL constructs to particular relations. We instantiate these constructs according to concept identifiers stored at lower levels and finally encode the respective μ-measure values (if they are present) using the special (F)OWL[2] constructs presented in Appendix B.

2. The acronym stands for (Fuzzy)OWL.

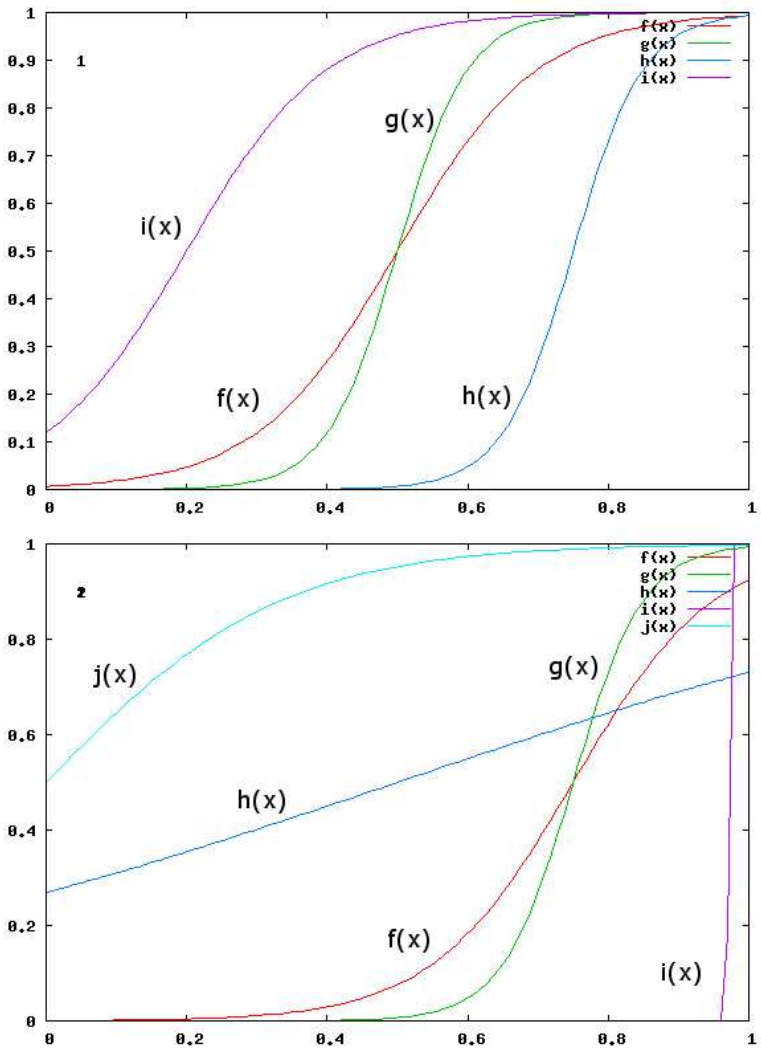


Figure 5.1: Examples of various shapes of the conviction function

... In the fairy book there is a lot of information on tritons, $\mathbf{mermaids}_{c_1}$, sea snakes and other $\mathbf{mythical\ creatures}_{c_2}$...
... $\mathbf{Mermaids}_{c_1}$ are considered as $\mathbf{females}_{c_3}$...
... for $\mathbf{sylphs}_{c_4}$, especially $\mathbf{mermaids}_{c_1}$, are banned to interfere with humans...

Figure 5.2: The text with concepts to be merged

Chapter 6

OLE Architecture

Technical issues regarding the OLE implementation are discussed in this chapter. First section overviews general demands on the software tools design, the second one presents the OLE system architecture and the last one deals with the implementation paradigm and external tools that are utilised within OLE.

6.1 Design Considerations

The design of OLE has been influenced by the need for autonomy, efficiency and precision of the resulting platform. The following list summarises the major requirements:

- The tool should support interactive way of ontology acquisition, but also the fully automatic process of knowledge mining that can run without any human assistance.
- The efficiency of ontology acquisition is crucial, for the system will process gigabytes of data.
- The precision is preferred over the recall. Even if the number of the extracted conceptual structures will be relatively low (compared to the number of relations a human can identify in the same resource), it will be balanced by the extensive quantity of resources available.
- The relations between concepts stored in the resulting ontology need not be precise — the explicit uncertain knowledge representation is one of the essential parts of OLE reasoning tools to be devised. The increased fuzzy precision of the whole process will balance the loss of exactness.

6.2 System Components

The modular architecture of OLE is given in Figure 6.1.

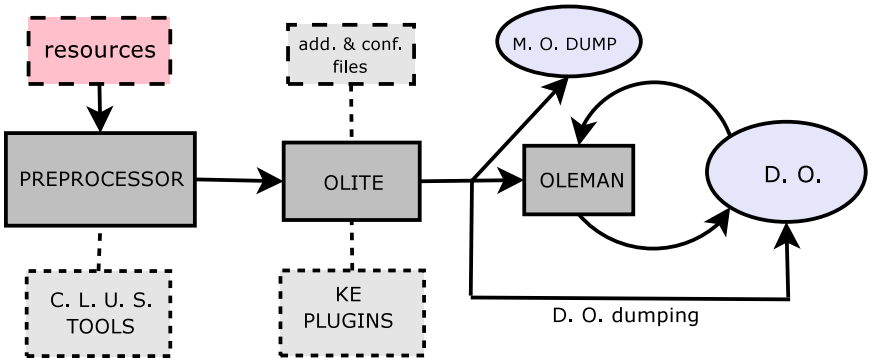


Figure 6.1: The architecture of the OLE platform

The **OLE** modules process plain text and create the miniontologies from the extracted data. Miniontologies can be directly dumped then or passed to the OLEMAN integration module. Here are comments on the figure above:

Resources — relevant documents provided by external tools (document classifiers, existing databases of related resources etc.).

C. L. U. S. Tools — cross–language universality support tools that allow OLE to preprocess data in a given language under specific conditions.

OLITE — the core of the extraction module responsible for creation of ontologies according to provided preprocessed data, utilising some of the knowledge extraction plugins. The ontologies are primarily produced in the internal $\mathcal{ANUIC}$ format (see Chapter 5), but they can be translated into the OWL format as well.

KE Plugins — knowledge extraction plugins implementing various methods for miniontology or even whole domain ontology generation.

Add. & Conf. Files — additional and configuration files (such as PES pattern representation file).

M. O. Dump — direct dump of miniontology as a product of OLITE module.

OLEMAN — module that merges the miniontologies resulting from the OLITE module and updates the base domain ontology. Basic method

described in Section 4.3 has been implemented so far. However, techniques of uncertain information representation will be employed further for the phase of ontology merging.

D. O. — continuously updated domain ontology.

More detailed descriptions of particular parts depicted in the list above are given mainly in Section 4, Section 4.1.1 and Section 4.3.

6.3 Implementation Remarks

All the OLE software components are implemented in the Python programming language. A special attention has been paid to the object oriented design. Another reason for choosing Python was the wide range of freely available relevant modules and application interfaces.

For example, NLTK natural language toolkit [1] and Princeton WordNet interface PyWordNet [2] are used. Besides the fact that it is programmed in Python, we use the NLTK toolkit instead of ready-made platforms (such as GATE, see [10]) mainly due to implementation of our own custom preprocessing tools. This approach allows us to port the system easily for different languages, not only English.

Python as an interpreted language can be inefficient for the implementation of some parts of the OLE platform. Special tools improving the computational efficiency of the Python code are available. For example, we are going to take advantage of Psyco [36] which is similar to the Java just-in-time compiler. Moreover, the computationally demanding parts can be straightforwardly implemented as C extensions for the main Python modules.

Chapter 7

Evaluation and Preliminary Results

In the following sections we present preliminary results of the ontology extraction and merging techniques that are used in OLE and described in this work. The chapter is divided into three sections that cover pattern–based extraction technique, method grounded in hierarchical clustering and basic merging of ontologies.

The evaluation of ontology is a problematic task even for hand–crafted ontologies. As stated for example in [6], the well known notions of precision and recall cannot be easily used. Following the source cited, we would like the *precision* to reflect the amount of knowledge correctly represented in an ontology with respect to all knowledge in ontology. The *recall* should reflect the amount of knowledge stored in an ontology with respect to all available knowledge then. It is obviously very hard to define a *correctly represented knowledge*, as well as to decide automatically what is *all available knowledge*. Therefore we cannot perform an exact and exhaustive evaluation similar for example to evaluation techniques used for POS tagging or disambiguation NLP tasks. We could use qualitative measures introduced for example in [44]. Theses measures reflect branching of relations, balance of relation trees in an ontology etc. However, these measures are devised mainly for manually created ontologies, whereas automatically gained OLE ontologies present only one kind of basic taxonomic relation. So it is not very useful to use the devised measures as an objective quality factor.

Due to the complications mentioned in the previous paragraph, we manually analysed representative or illustrational samples of ontologies having been gained from various (computer science related) resources. The presented values of precision are orientational ratios of number of relations that were found to be ”reasonable” compared to number of all relations in an ontology. The recall values are even more difficult to be specified as we further develop in the following sections.

7.1 Pattern-based Method

We tested the OLITE pattern–based extraction module on a set of about $12,000$ documents from computer science domain. We used the hand–crafted patterns (some of them were adopted according to [24] and [15]) that are given in Table 7.1. Other patterns can be added easily, but the patterns

Id	The pattern
1	NP such as (NPList | NP)
2	such NP as (NPList | NP)
3	(NPList | NP) (and | or)other NP
4	NP (including | especially) (NPList | NP)
5	(NPList | NP) (is | was)an? NP
6	(NPList | NP) is the NP
7	(NPList | NP) and similar NP
8	NP like (NPList | NP)

Table 7.1: Patterns for *is-a* relation

presented in the table were found to be sufficient for basic evaluation.

The average time of processing of a resource (only extraction without miniontology dumping or merging) was 0.94 seconds[1]. Average size of a document in the sample that was used for this performance test was about 66.7 kilobytes (approximately $9,500$ words). This results in a processing speed of about $10,100$ words per second which we find satisfactory.

For the manual evaluation we randomly chose ten resources from the whole document set. For each miniontology created by OLE system we computed the ratio of ”reasonable” relations compared to all extracted relations[2]. Reasonability of a relation was decided with respect to the corresponding resource. This ratio presents an orientational measure of the *precision*. As an approximate measure of the *recall* we chose the ratio of sentences that matched with the patterns compared to all sentences in a resource. This is not an exact measure of recall with respect to all knowledge present in a resource, but it can give us a glimpse of how the data in a resource are covered by the pattern–based module. All of the measures mentioned here are summed up in Table 7.2 and provided with respective file size and num-

1. On a machine with 3.2 GHz Intel Pentium 4 processor and 2GB of RAM, powered by Ubuntu Linux operating system.
2. For simplicity, we consider the relations for a concept in common, defining a relation ”unreasonable” if any of the respective relations has not been found to be reasonable.

ber of all concepts extracted. See Appendix A for concrete examples of extracted taxonomies.

File	File sz. (words)	No. of conc.	No. of rel.	Prec. (%)	Rec. (%)	I (%)
1	3330	7	5	60.00	23.52	840.34
2	2606	9	5	80.00	5.21	1438.85
3	5387	33	24	62.50	5.88	4401.41
4	2274	16	11	63.63	3.31	2179.11
5	3936	25	14	71.43	7.51	4277.25
6	4943	27	18	61.11	5.84	3892.36
7	3937	22	15	46.67	4.27	3070.39
8	7438	25	16	68.75	7.37	3756.83
9	1826	10	5	60.00	6.19	1801.80
10	5250	52	32	37.50	18.42	8333.33
average	4093	22.6	14.5	61.16	8.75	3399.17

Table 7.2: Results of pattern–based extraction

The precision values are quite high when we look at the I column in the table. The I values present an improvement in precision over a base–line, which is computed as $\frac{R_R}{N(N-1)}$, where R_R stands for number of reasonable relations and N is the number of concepts in an ontology[3]. Moreover, it is precision of the extraction phase and it would be significantly improved by utilisation of a proper reasoning engine. This is one of the main goals of our future work.

7.2 CAANNO Technique

The CAANNO algorithm hierarchically clusters all single noun terms in a resource and then assigns a common label for each class in the resulting dendrogram. We do not define either formal or informal measure of recall this time, because all identified nouns in a resource set are clustered. However, we use the same notion of precision as was used in the previous section.

Due to the strenuousness of manual evaluation of large ontologies we used only a set of 131 concepts (non–unique individuals) from a coherent computer science domain resource. 62 unique individuals and 47 classes were induced. We distinguished between *class–class* relationships and *class–*

3. The $N(N-1)$ is number of all *is-a* relations that can be assigned among all concepts.

individual relationships when analysing the precision. The precision values are given in Table 7.3, supplemented by the number of respective relations.

Rel. kind	No. of relations	Precision (%)
class–class	47	44.68
class–individual	62	51.61
average	54.5	48.15

Table 7.3: Results of CAANNO extraction

The precision is lower than for the pattern–based method, but it slightly increases with more data. Moreover, the absolute recall is no doubt much higher. Relatively low precision could also be improved by utilisation of reasoning supported by integration of more precise ontologies and even by heuristic comparison of different cuts of a dendrogram for the same resource set.

7.3 Basic Ontology Merging

The method used for ontology merging is provisional in the scope of future OLE development. Therefore we performed only very simple evaluation of the technique described in this work. We took the domain ontology with size of 62 individuals and 47 classes and integrated it with miniontologies with common size of 44 concepts (classes and implicit individuals). New ontology has 88 individuals and 60 classes. Table 7.4 shows the orientational precision of implemented merging. The informal precision P_{merged} is computed as $P_{merged} = \frac{Prec_{upd}}{Prec_{orig}}$, where $Prec_{upd}$ and $Prec_{orig}$ are the precisions assigned to updated and original domain ontology as defined in the previous two sections[4]. We consider the obtained precision values as a

Rel. kind	$Prec_{orig}$ (%)	$Prec_{upd}$ (%)	$P_{merged}(\%)$
class–class	44.68	40.98	91.72
class–individual	51.61	49.44	95.80
average	48.15	45.21	93.76

Table 7.4: Results of merging

4. The recall measure is of no significance again, because we merge all relations present in the input ontologies.

good foundation of the base–line definition for devising and evaluation of improved ontology merging techniques.

Chapter 8

Conclusions and Future Work

We presented OLE — a novel ontology learning platform in this work. OLE is primarily intended for autonomous creation and management of domain specific ontologies. The bottom–up approach to the ontology acquisition is emphasised, as well as the need for uncertainty representation. The OLITE component implements two basic knowledge acquisition methods, First of them is pattern–based extraction. The second one is an original technique CAANNO that utilises automatic hierarchical annotation of word clusters using WordNet lexical database. Other modules can be easily added. Simple heuristic ontology merging technique was introduced as well.

The preliminary results clearly show that OLE provides a modular and flexible platform for integration of knowledge extraction techniques. Moreover, the results also prove that the purely automatic construction of knowledge bases of quite a reasonable quality is more than feasible.

We also presented an $\mathcal{ANUIC}$ framework for natural dealing with uncertain knowledge in ontologies. The framework is motivated by intuitive, yet valuable notion of representation of uncertainty in a human mind. The theoretical background of fuzzy sets methodology allows to develop an appropriate calculus and consecutively build inference tools to reason among the concepts stored in $\mathcal{ANUIC}$.

The research results presented here are mostly in the phase of proposal and proof of concept, so a lot of work still has to be done in order to acquire really complex ontologies. An elaborate interface with server–client architecture should be build for OLE tools. Such an interface should be user–friendly when considering both human and artificial agents. Challenging work remains to be done in the area of dynamic acquisition of new patterns and extraction techniques that are not limited only to *is-a* relation. Many advanced techniques for concept mining still wait for their implementation. The ontology merging process in OLE will benefit from significantly increased coverage implied by other extraction approaches as well as from the automatic induction of semantic relation patterns.

The second big objective for future work is to find efficient implementa-

tion methods for the proposed fuzzy ontologies. Additional psycholinguistic experiments should help with proper configuration of the parameters for the μ-measure function presented in Section 5.3 then. Invention and formal validation of a specific calculus for $\mathcal{ANUIC}$ is also needed. Then we can evaluate the framework using real world data from distinct domains of OLE project. All of the mentioned tasks are no doubt difficult, but we demand it would be very challenging to pursue them and refine the ideas behind to gain a sustainable and efficient universal model of representation and processing of uncertain knowledge. Thus we can make our automatically gained OLE ontologies as complex and empirically valid as possible within the future research.

Acknowledgements

This work has been supported by Academy of Sciences of Czech Republic, 'Information Society' program, the national research project 1ET100300419.

Bibliography

[1] *NLTK: Natural Language Toolkit – Technical Reports*, 2005. Available at: http://nltk.sourceforge.net/tech/index.html.

[2] *PyWordNet: Python Interface to Princeton WordNet*, 2005. Available at: http://osteele.com/projects/pywordnet/.

[3] J. Allwood. Meaning potentials and context: Some consequences for the analysis of variation and meaning. In *Cognitive Approaches to Lexical Semantics*, pages 29–66. Mouton de Gruyter, Berlin, 2003.

[4] J. C. Arpirez, O. Corcho, M. Fernandez-Lopez, and A. Gomez-Perez. Webode in a nutshell. *AI Magazine*, 24(3):37–47, 2003.

[5] X. Li B. Wang, W. Liu, and Y. Shi. A new sparse rule-based fuzzy reasoning method. In *Proceedings of the Fourth International Conference on Hybrid Intelligent Systems (HIS'04)*, pages 462–467. IEEE Computer Society, 2004.

[6] C. Brewster, H. Alani, S. Dasmahapatra, and Y. Wilks. Data driven ontology evaluation. In *Proceedings of LREC 2004*, 2004.

[7] E. Brill. A report of recent progress in transformation-based error-driven learning. In *Proc. ARPA Human Language Technology Workshop '94*, Princeton, NJ, 1994.

[8] O. Corcho, A. Gomez-Perez, A. Lopez-Cima, and V. Lopez. Odesew. automatic generation of knowledge portals for intranets and extranets. In *Proceedings of International Conference on The Semantic Web — ISWC 2003*, pages 802–817, Berlin Heidelberg, 2003. Springer–Verlag.

[9] C. Cornelis and E. Kerre. Inclusion measures in intuitionistic fuzzy set theory. In *ECSQARU '03: Proceedings of the 7th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 345–356, London, UK, 2003. Springer-Verlag.

[10] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, 2002.

[11] H. Cuyckens, R. Dirven, and J. R. Taylor, editors. *Cognitive Approaches to Lexical Semantics*, volume 23. Mouton de Gruyter, Berlin, cognitive linguistics research edition, 2003.

[12] J. Derrida. *A Derrida Reader: between the Blinds*. Harvester Wheatsheaf, New York, 1991.

[13] D. Dubois and H. Prade. Rough fuzzy sets and fuzzy rough sets. *International Journal of General Systems*, 17:191–209, 1990.

[14] I. Düntsch. A logic for rough sets. *Theoretical Computer Science*, 179(1-2):427–436, 1997.

[15] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Web-scale information extraction in knowitall: (preliminary results). In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 100–110, New York, NY, USA, 2004. ACM Press.

[16] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[17] R. Fikes. Ontologies: What are they, and where's the research? In *KR'96: Principles of Know ledge Representation and Reasoning*, pages 652–654. Morgan Kaufmann, San Francisco, California, 1996.

[18] A. Gangemi, R. Navigli, and P. Velardi. Corpus driven ontology learning: a method and its application to automated terminology translation. *IEEE Intelligent Systems*, pages 22–31, 2003.

[19] L. Garmendia and A. Salvador. Computing a transitive opening of a reflexive and symmetric fuzzy relation. In *ECSQARU '05: Proceedings of the 7th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 587–599, London, UK, 2005. Springer-Verlag.

[20] M. R. Genesereth. *Knowledge Interchange Format: draft proposed American National Standard (dpANS), NCITS.T2/98-004*. Available at:
http://logic.stanford.edu/kif/dpans.html.

[21] R. Giugno and T. Lukasiewicz. P-$\mathcal{SHOQ}$(D): A probabilistic extension of $\mathcal{SHOQ}$(D) for probabilistic ontologies in the semantic web. In *JELIA '02: Proceedings of the European Conference on Logics in Artificial Intelligence*, pages 86–97, London, UK, 2002. Springer-Verlag.

[22] T. R. Gruber. A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199–220, 1993.

[23] C. M. Teng H. E. Kyburg, J. Kyburg. *Uncertain Inference*. Cambridge University Press, Cambridge, 2001.

[24] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics*, pages 539–545, Morristown, NJ, USA, 1992. Association for Computational Linguistics.

[25] M. A. Hearst and E. Stoica. Nearly–automated metadata hierarchy creation. In *Proceedings of HLT-NAACL'04*, 2004.

[26] D. Hofstadter. *Fluid Concepts & Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought*. Basic Books, New York, 1995.

[27] M. Holi and E. Hyvönen. A method for modeling uncertainty in semantic web taxonomies. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 296–297, New York, NY, USA, 2004. ACM Press.

[28] G. J. Klir and M. J. Wierman. *Uncertainty-Based Information: Elements of Generalized Information Theory*. Physica-Verlag/Springer-Verlag, Heidelberg and New York, 1999.

[29] H. Knublauch. Ontology driven software development in the context of the semantic web: An example, scenario with protégé/owl. In *Proceedings of 1st International Workshop on the Model-Driven Semantic Web (MDSW2004)*, 2004.

[30] T. B. Lee, D. R. Karger, L. A. Stein, R. R. Swick, and D. J. Weitzner. *Semantic Web Development - Technical Proposal*, 2000. Available at: http://www.w3.org/2000/01/sw/DevelopmentProposal.

[31] P. Materna. Rehabilitation of concepts. Available at:
http://www.phil.muni.cz/~materna/rehabilitation_of_concepts.html.

[32] R. Meersman. Ontologies and databases: More than a fleeting resemblance, 2001.

[33] J. L. Old. The semantic structure of roget's, a whole-language thesaurus, 2003.

[34] E. Hovy P. Pantel, D. Ravichandran. Towards terascale knowledge acquisition. In *Proceedings of Conference on Computational Linguistics (COLING-04)*, pages 771–777, 2004.

[35] P. Pantel and D. Lin. Discovering word senses from text, 2002.

[36] A. Rigo. *The Ultimate Psyco Guide*, 2005. Available at: http://psyco.sourceforge.net/psycoguide/index.html.

[37] M.-C. Rousset. Small can be beautiful in the semantic web. In *ISWC 2004: Third International Semantic Web Conference. Proceedings*, pages 6–16. Springer-Verlag Berlin Heidelberg, 2004.

[38] P. Rychlý and P. Smrž. Manatee, bonito and word sketches for czech. In *Proceedings of the Second International Conference on Corpus Linguisitcs*, pages 124–132. Saint-Petersburg State University Press, 2004.

[39] A. Silberschatz, H. F. Korth, and S. Sudershan. *Database System Concepts*. McGraw-Hill, Inc., New York, NY, USA, 1998.

[40] J. F. Sowa. Conceptual graphs for a data base interface. *IBM Journal of Research and Development*, 20(4):336–357, 1976.

[41] J. F. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing Co., 2000.

[42] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. Ontoedit: Collaborative ontology development for the semantic web. In *ISWC 2002: First International Semantic Web Conference. Proceedings*, pages 221–235. Springer-Verlag Berlin Heidelberg, 2002.

[43] T. H. Cao T. T. Quan, S. C. Hui. Automatic generation of ontology for scholarly semantic web. In *ISWC 2004: Third International Semantic Web Conference. Proceedings*, pages 726–740. Springer-Verlag Berlin Heidelberg, 2004.

[44] S. Tartir, I. B. Arpinar, M. Moore, A. P. Sheth, and B. Aleman-Meza. Ontoqa: Metric-based ontology quality analysis. In *Proceedings of IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources*, 2005.

[45] A. Ushioda. Hierarchical clustering of words. In *Proceedings of the 16th conference on Computational linguistics*, pages 1159–1162, Morristown, NJ, USA, 1996. Association for Computational Linguistics.

[46] P. Vossen, editor. *EuroWordNet: A Multilingual Database with Lexical Semantic Networks*. Kluwer, Dordrecht, Netherlands, 1998.

[47] Y. Xiang. *Probabilistic Reasoning in Multi-agent Systems: A Graphical Models Approach*. Cambridge University Press, Cambridge, 2002.

[48] R. Pan Y. Peng, Z. Ding. Bayesowl: A probabilistic framework for uncertainty in semantic web. In *Proceedings of Nineteenth International Joint Conference on Artificial Intelligence (IJCAI05)*, 2005.

[49] L. A. Zadeh. Fuzzy sets. *Journal of Information and Control*, 8:338–353, 1965.

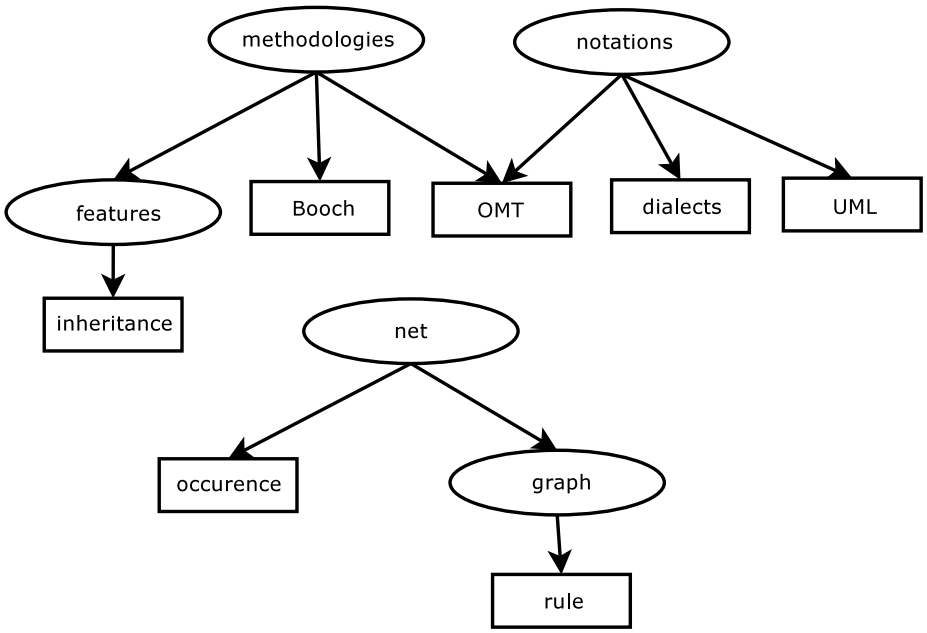
Appendix A

OLE Ontologies — Basic Real Data Examples

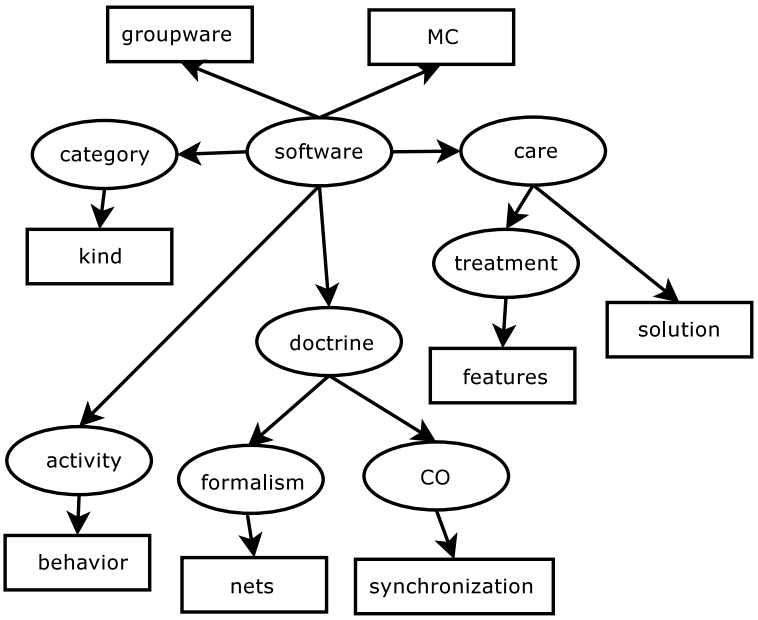
The figures below represent illustrational samples from ontologies created by OLE. We use the following graphic notation:

- classes are depicted as ellipses;
- individuals as squares;
- and arrows from hyperonyms to hyponyms encode the *is-a* relation.

A.1 Sample from an Ontology Extracted Using Patterns



A.2 Sample from an Ontology Extracted by CAANNO



Appendix B

(F)OWL — Fuzzy OWL Extension

```
<?xml version="1.0"?>
<!--
Created on Dec. 12, 2005 by Vit Novacek
Defining Fuzzy Extension Markups
Example:
The class referred by ID=135875 is a class referred by ID=869723
with conviction 0.9235...
<Conviction rdf:ID="is_a(135875,869723)">
<hasRelation>isa</hasRelation>
<hasClass>135875</hasClass>
<hasClass>869723</hasClass>
<hasConvValue>0.9235</hasConvValue>
</Conviction>
-->

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns="http://nlp.fi.muni.cz/projects/ole/owl/fuzzy.owl#"
>
<owl:Ontology rdf:about="http://nlp.fi.muni.cz/projects/ole/owl/
fuzzy.owl#">
<owl:versionInfo>v1.0</owl:versionInfo>
</owl:Ontology>

<owl:Class rdf:ID="Conviction">
<owl:unionOf rdf:parseType="Collection">
<owl:Class>
<rdfs:subClassOf>
```

```
<owl:Restriction>
<owl:onProperty rdf:resource="#hasClass"/>
<owl:cardinality rdf:datatype="http://www.w3.org/2001/
XMLSchema#nonNegativeInteger">1</owl:cardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasClass"/>
<owl:cardinality rdf:datatype="http://www.w3.org/2001/
XMLSchema#nonNegativeInteger">1</owl:cardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasValue"/>
<owl:cardinality rdf:datatype="http://www.w3.org/2001/
XMLSchema#nonNegativeInteger">1</owl:cardinality>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
<owl:Class>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasIndividual"/>
<owl:cardinality rdf:datatype="http://www.w3.org/2001/
XMLSchema#nonNegativeInteger">1</owl:cardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasClass"/>
<owl:cardinality rdf:datatype="http://www.w3.org/2001/
XMLSchema#nonNegativeInteger">1</owl:cardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasValue"/>
<owl:cardinality rdf:datatype="http://www.w3.org/2001/
```

```
XMLSchema#nonNegativeInteger">1</owl:cardinality>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
<owl:Class>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasClass"/>
<owl:cardinality rdf:datatype="http://www.w3.org/2001/
XMLSchema#nonNegativeInteger">1</owl:cardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasIndividual"/>
<owl:cardinality rdf:datatype="http://www.w3.org/2001/
XMLSchema#nonNegativeInteger">1</owl:cardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasValue"/>
<owl:cardinality rdf:datatype="http://www.w3.org/2001/
XMLSchema#nonNegativeInteger">1</owl:cardinality>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
<owl:Class>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasIndividual"/>
<owl:cardinality rdf:datatype="http://www.w3.org/2001/
XMLSchema#nonNegativeInteger">1</owl:cardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasIndividual"/>
<owl:cardinality rdf:datatype="http://www.w3.org/2001/
XMLSchema#nonNegativeInteger">1</owl:cardinality>
```

```
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasValue"/>
<owl:cardinality rdf:datatype="http://www.w3.org/2001/
XMLSchema#nonNegativeInteger">1</owl:cardinality>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
</owl:unionOf>
</owl:Class>

<owl:ObjectProperty rdf:ID="hasRelation">
<rdfs:domain rdf:resource="#Conviction"/>
<rdfs:range rdf:resource="http://www.w3.org/2002/07/owl#
ObjectProperty"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasClass">
<rdfs:domain rdf:resource="#Conviction"/>
<rdfs:range rdf:resource="http://www.w3.org/2002/07/owl#
Class"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasIndividual">
<rdfs:domain rdf:resource="#Conviction"/>
<rdfs:range rdf:resource="http://www.w3.org/2000/01/
rdf-schema#Literal"/>
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:ID="hasConvValue">
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
FunctionalProperty"/>
<rdfs:domain rdf:resource="#Conviction"/>
<rdfs:range rdf:resource="http://nlp.fi.muni.cz/projects/
/ole/owl/dt.xsd#between0and1"/>
</owl:DatatypeProperty>

</rdf:RDF>
```