Learning Expressive Ontologies

Johanna VÖLKER [1] and Peter HAASE and Pascal HITZLER
Institute AIFB, University of Karlsruhe, Germany

Abstract. The automatic extraction of ontologies from text and lexical resources has become more and more mature. Nowadays, the results of state-of-the-art ontology learning methods are already good enough for many practical applications. However, most of them aim at generating rather inexpressive ontologies, i.e. bare taxonomies and relationships, whereas many reasoning-based applications in domains such as bioinformatics or medicine rely on much more complex axiomatizations. Those are extremely expensive if built by purely manual efforts, and methods for the automatic or semi-automatic construction of expressive ontologies could help to overcome the knowledge acquisition bottleneck. At the same time, a tight integration with ontology evaluation and debugging approaches is required to reduce the amount of manual post-processing which becomes harder the more complex learned ontologies are. Particularly, the treatment of logical inconsistencies, mostly neglected by existing ontology learning frameworks, becomes a great challenge as soon as we start to learn huge and expressive axiomatizations. In this chapter we present several approaches for the automatic generation of expressive ontologies along with a detailed discussion of the key problems and challenges in learning complex OWL ontologies. We also suggest ways to handle different types of inconsistencies in learned ontologies, and conclude with a visionary outlook to future ontology learning and engineering environments.

Keywords. Ontology Learning, Reasoning, Ontology Evolution, Ontology Engineering

1. Introduction

During the last decade ontologies have become an important means for knowledge interchange and integration. They are used for corporate knowledge management, web portals and communities, semantic search, and web services. A couple of ontology languages have emerged as wide-spread means for ontology representation - among them the web ontology language, OWL, which provides a powerful formalism for knowledge representation and reasoning. OWL was proposed as a world-wide standard by the W3C committee, and several subsets of the OWL language with different expressivity have been defined in order to meet the demands of a great variety of semantic applications, and of course the great vision of the semantic web.

However, the realization of the semantic web as envisioned by Tim-Berners Lee is still hampered by the lack of ontological resources. Building ontologies is a difficult and time-consuming task. It usually requires to combine the knowledge of domain experts

[1]Corresponding Author: Johanna Völker, Institut AIFB, Universität Karlsruhe (TH), 76128 Karlsruhe, Germany; E-mail: voelker@aifb.uni-karlsruhe.de.

with the skills and experience of ontology engineers into a single effort with high demand on scarce expert resources. We believe that this bottleneck currently constitutes a severe obstacle for the transfer of semantic technologies into practice.

In order to address this bottleneck, it is reasonable to draw on available data, applying automated analyses to create ontological knowledge from given resources or to assist ontology engineers and domain experts by semi-automatic means. Accordingly, a significant number of ontology learning tools and frameworks has been developed aiming at the automatic or semi-automatic construction of ontologies from structured, unstructured or semi-structured documents. The current state-of-the-art in lexical ontology learning is able to generate ontologies that are largely *informal* or *lightweight* ontologies in the sense that they are limited in their expressiveness and often only consist of concepts organized in a hierarchy. While less expressive, informal ontologies have proven useful in certain application scenarios – an observation that also resonates with the so-called Hendler hypothesis [1]: "A little semantics goes a long way." – more and more people tend to see the future of semantic technologies in application scenarios such as e-business or bio-informatics which require large scale reasoning over complex domains. These knowledge-intensive applications even more than the semantic web depend on the availability of expressive, high-quality ontologies. However, both quality and expressivity of the ontologies which can be generated by the state-of-the-art ontology learning systems fail to meet the expectations of people who argue in favor of powerful, knowledge-intensive applications based on ontological reasoning. While it might seem infeasible to improve upon both at the same time, we argue that learning more expressive ontologies (e.g. by adding disjointness axioms) does not only yield sufficiently good results, but may also help in the task of automatic ontology evaluation, thus improving the overall quality of learned ontologies.

In this chapter, we present two complementary approaches to the automatic generation of expressive ontologies suitable for reasoning-based applications. The first approach is essentially based on a syntactic transformation of natural language definitions into description logic axioms. It hinges critically on the availability of sentences which have definitory character, like "*Enzymes are proteins that catalyse chemical reactions.*" Such sentences could be obtained e.g. from glossaries or software documentation related to the underlying ontology-based application. We exemplify this approach with definitions taken from a fishery glossary used in a case study at the *Food and Agriculture Organization of the United Nations* (FAO).

The second approach relies on a machine learning classifier for determining disjointness of any two classes. For its implementation, we developed a variety of different methods to automatically extract lexical and logical features which we believe to provide a solid basis for learning disjointness. These methods take into account the structure of the ontology, associated textual resources, and other types of data sources in order to compute the likeliness of two classes to be disjoint. The features obtained from these methods are used to build an overall classification model which we evaluated against a set of manually created disjointness axioms.

The support for reasoning is the major benefit of the use of expressive ontologies grounded in logics. Reasoning can be used in different phases of the lifecycle of an ontology. At runtime, reasoning allows to derive conclusions from the ontology, e.g. for the purpose of query answering over the ontology. At development time, reasoning can be used to validate the ontology and check whether it is non-contradictory, i.e. free of log-

ical inconsistencies. The more expressive the ontology language, the more precisely the intended meaning of a vocabulary can be specified, and consequently, the more precise conclusions can be drawn. Introducing disjointness axioms, for example, greatly facilitates consistency checking and the automatic evaluation of individuals in a knowledge base with regards to a given ontology.

Another particularly important topic for ontology learning is the challenge of dealing with inconsistencies. The reason lies in the fact that all ontology learning approaches generate knowledge that is afflicted with various forms of imperfection. The causes of imperfection may already be in the data sources from which the ontologies are generated, or they may be introduced by the ontology learning algorithms. To address this problem, we illustrate in this chapter how ontology learning can be combined with *consistent ontology evolution*, a reasoning-supported process to guarantee that the learned ontologies are kept consistent as the ontology learning procedures generate changes to the ontology over time.

The chapter is structured as follows. In the subsequent section we provide a brief introduction to the ontology language OWL and the main reasoning tasks in OWL as a Description Logic. In Section 3 we present ontology learning methods for learning expressive ontologies, particularly focusing on learning complex concept descriptions and disjointness axioms as two expressive language elements. In Section 4 we critically discuss the current state-of-the-art in learning expressive ontologies, analyzing problems and open issues. In Section 5 we show how inconsistencies can be dealt with in the context of ontology learning to guarantee a consistent evolution of the learned ontologies. We then propose a way of integrating ontology learning and evolution into the ontology lifecycle in Section 6. In Section 7 we present experiments in a concrete application scenario in the domain of fishery ontologies before concluding in Section 8.

2. OWL Ontologies and Reasoning Tasks

Traditionally, a number of different knowledge representation paradigms have competed to provide languages for representing ontologies, including most notably description logics and frame logics. With the advent of the OWL Web Ontology Language, developed by the Web Ontology Working Group and recommended by the World Wide Web Consortium (W3C), a standard for the representation of ontologies has been created. Adhering to this standard, we base our work on the OWL language (in particular OWL DL, as discussed below) and describe the developed formalisms in its terms.

2.1. OWL as a Description Logic

The OWL ontology language is based description logics, a family of class-based knowledge representation formalisms. In description logics, the important notions of a domain are described by means of concept descriptions that are built from *concepts* (also referred to as *classes*), *roles* (also referred to as *properties* or *relations*), denoting relationships between things, and *individuals* (also referred to as *instances*). It is now possible to state facts about the domain in the form of axioms. Terminological axioms make statements about how concepts or roles are related to each other, assertional axioms (sometimes also called *facts*) make statements about the properties of individuals of the domain.

We here informally introduce the language constructs of the description logic $\mathcal{SHOIN}$, the description logic underlying OWL DL. For the correspondence between our notation and various OWL DL syntaxes, see [2]. In the description logic $\mathcal{SHOIN}$, we can build complex classes from atomic ones using the following constructors:

- $C \sqcap D$ (intersection), denoting the concept of individuals that belong to both C and D,
- $C \sqcup D$ (union), denoting the concept of individuals that belong to either C or D,
- $\neg C$ (complement), denoting the concept of individuals that do not belong to C,
- $\forall R.C$ (universal restriction), denoting the concept of individuals that are related via the role R only with individuals belonging to the concept C,
- $\exists R.C$ (existential restriction), denoting the concept of individuals that are related via the role R with some individual belonging to the concept C,
- $\geq n\, R$, $\leq n\, R$ (qualified number restriction), denoting the concept of individuals that are related with at least (at most) n individuals via the role R.
- $\{c_1, \ldots, c_n\}$ (enumeration), denoting the concept of individuals explicitly enumerated.

Based on these class descriptions, axioms of the following types can be formed:

- concept inclusion axioms $C \sqsubseteq D$, stating that the concept C is a subconcept of the concept D,
- transitivity axioms $\mathsf{Trans}(R)$, stating that the role R is transitive,
- role inclusion axioms $R \sqsubseteq S$ stating that the role R is a subrole of the role S,
- concept assertions $C(a)$ stating that the individual a is in the extension of the concept C,
- role assertions $R(a, b)$ stating that the individuals a, b are in the extension of the role R,
- individual (in)equalities $a \approx b$, and $a \not\approx b$, respectively, stating that a and b denote the same (different) individuals.

Using the constructs above, we can make complex statements, e.g. expressing that two concepts are disjoint with the axiom $A \sqsubseteq \neg B$. This axioms literally states that A is a subconcept of the complement of B, which intuitively means that there must not be any overlap in the extensions of A and B.

In the design of description logics, emphasis is put on retaining decidability of key reasoning problems and the provision of sound and complete reasoning algorithms. As the name suggests, Description Logics are logics, i.e. they are formal logics with well-defined semantics. Typically, the semantics of a description logic is specified via *model theoretic semantics*, which explicates the relationship between the language syntax and the models of a domain.

An interpretation consists of a domain of interpretation (essentially, a set) and an interpretation function which maps from individuals, concepts and roles to elements, subsets and binary relations on the domain of interpretation, respectively. A description logic knowledge base consists of a set of axioms which act as constraints on the interpretations. The meaning of a knowledge base derives from features and relationships that are common in all possible interpretations. An interpretation is said *to satisfy a knowledge base*, if it satisfies each axiom in the knowledge base. Such an interpretation is called a *model* of the knowledge base. If there are no models, the knowledge base is said to be

inconsistent. If the relationship specified by some axiom (which may not be part of the knowledge base) holds in all models of a knowledge base, the axiom is said to be *entailed* by the knowledge base. Checking consistency and entailment are two standard reasoning tasks for description logics. Other reasoning tasks include computing the concept hierarchy and answering conjunctive queries.

2.2. Approaches to Dealing with Inconsistencies

Standard entailment as defined above is explosive, i.e. an inconsistent ontology has *all* axioms as consequences. Formally, if an ontology O is inconsistent, then for all axioms α we have $O \models \alpha$. In other words, query answers for inconsistent ontologies are completely meaningless, since for all queries the query answer will be *true*. To deal with the issue of potential inconsistencies in ontologies, we can choose from a number of alternative approaches [3]:

Consistent Ontology Evolution is the process of managing ontology changes by preserving the consistency of the ontology with respect to a given notion of consistency. The consistency of an ontology is defined in terms of consistency conditions, or invariants that must be satisfied by the ontology. The approach of consistent ontology evolution imposes certain requirements with respect to its applicability. For example, it requires that the ontology is consistent in the first place and that changes to the ontology can be controlled. In certain application scenarios, these requirements may not hold, and consequently, other means for dealing with inconsistencies in changing ontologies may be required.

Repairing Inconsistencies involves a process of diagnosis and repair: first the cause (or a set of potential causes) of the inconsistency needs to be determined, which can subsequently be repaired. Unlike the approach of consistent ontology evolution, repairing inconsistencies does not require to start with a consistent ontology and is thus adequate if the ontology is already inconsistent in the first place.

Reasoning with Inconsistent Ontologies does not try to avoid or repair the inconsistency (as in the previous two approaches), but simply tries to "live with it" by trying to return meaningful answers to queries, even though the ontology is inconsistent. In some cases consistency cannot be guaranteed at all and inconsistencies cannot be repaired, still one wants to derive meaningful answers when reasoning.

3. Learning Expressive OWL Ontologies

In the following we propose two complementary approaches to support the generation of expressive ontologies suitable for reasoning-based applications. After a brief overview of state-of-the-art methods for ontology learning we first present LExO, a prototypical implementation supporting the automatic generation of complex class descriptions from lexical resources (cf. Section 3.2). In Section 3.3, we focus on the task of creating disjointness axioms, and describe a classification-based approach using a combination of lexical and logical features for capturing disjointness.

3.1. Learning Ontology Elements and Basic Axioms

Ontology learning so far has focussed on the extraction of ontology elements such as concepts, instances or relations, as well as simple axioms. In this section, we briefly present some of the most frequently used methods for generating these types of primitives (for a more complete survey see, e.g. [4]).

3.1.1. Ontology Elements

Concepts and Instances. Different term weighting measures such as TFIDF, relative term frequency, entropy or C-value / NC-value [5] are used for identifying those terms which are most relevant for the domain of interest. Whereas the domain is modeled by a given document corpus each of the extracted noun phrases is assumed to represent either a concept or an instance. The distinction between concepts and instances is typically made depending on the part-of-speech information associated with its lexical representation, i.e. common and proper nouns.

General Relations. Approaches based on subcategorization frames rely on the assumption that ontological relationships are mostly represented by verbs and their arguments. Accordingly, selectional restrictions usually reflect domain and range restrictions of these relations [6,7]. In this line, Navigli and Velardi [8] extract taxonomic and non-taxonomic, i.e. general relations from glossaries and thesauri. Their approach is based on regular expressions further restricted by syntactic and semantic constraints, as well as a word sense disambiguation component which links extracted terms to WordNet. A more general approach which also considers attributive descriptions of concepts has been developed by Poesio and Almuhareb [9] who evaluated the use of a machine learning classifier for selecting the most distinctive attributes and relations for each concept. Unlabeled relations can be extracted by association rules which try to capture the semantic correlation of two elements based on their co-occurrences in the corpus. Unlike relations expressed by verbal or attributive phrases these anonymous relations have to be labeled by the ontology engineer in a post-processing step [10]. In addition to these two kinds of approaches, a number of methods for discovering particular types of relationships, e.g. part-of relations [11], have been developed so far.

3.1.2. Axioms

Subclass-of Relations. Many approaches for learning subclass-of relationships exploit hyponymy information in WordNet, or rely on Hearst patterns [12] as indicators for hyponymy relationships. Moreover, methods based on hierarchical clustering [13,14,15] and Formal Concept Analysis [16] have been developed for inducing taxonomies by grouping concepts with similar lexical context. Lexical context in its simplest form consists of the weighted co-occurrences of a term, but it may also include any kind of syntactic dependencies such as predicates, or prepositional complements associated with a given term.

Instance-of Relations. Distributional similarity, i.e. similarity based on lexical context, can be considered as an indicator for certain paradigmatic relationships, which makes it a suitable means for identifying, e.g. concept instantiation. Consequently, approaches such as [17] assign instances to the semantically most similar class by computing the contextual overlap. Other approaches to learning instance-of relationships rely on manually engineered or automatically acquired lexico-syntactic patterns [18,19].

```
( E1 () (fin) C
   ( 3 is (be) VBE i
      ( 2 farmer (~) N s
         ( 1 A (~) Det det )
      )
      ( 5 person (~) N pred
         ( E3 () (farmer) N subj 2 )
         ( 4 a (~) Det det )
         ( E0 () (fin) C rel
            ( 6 who (~) N whn 5 )
            ( 7 operates (operate) V i
               ( E4 () (who) N subj 5 )
               ( 9 farm (~) N obj
                  ( 8 a (~) Det det )
) ) ) ) ) )
```

Figure 1. Dependency Tree (Minipar)

```
rule: relative clause {
    arg_0:  //N
    arg_1:  arg_0 /C[@role='rel']
    arg_2:  arg_1 /V
    result: [equivalent 0 [and 0-1 2]]
}
rule: verb and object {
    arg_0:  //V
    arg_1:  arg_0 /N[@role='obj']
    result: [equivalent 0 [some 0-1 1]]
    result: [subObjectPropertyOf 0 0-1]
}
```

Figure 3. Transformation Rules

3.2. Learning Class Descriptions

LExO[2] (Learning EXpressive Ontologies) [20] is among the first approaches towards the automatic generation of ontologies featuring the full expressiveness of OWL DL. The core of LExO is a syntactic transformation of definitory natural language sentences into description logic axioms.

Given a natural language definition of a class, LExO starts by analyzing the syntactic structure of the input sentence. The resulting dependency tree is then transformed into a set of OWL axioms by means of manually engineered transformation rules. In the following, we provide a step-by-step example to illustrate the complete transformation process. For more (and more complicated) examples please refer to Section 7.

3.2.1. Example

Here, we assume that we would like to refine the description of the class *Farmer* which could be part of an agriculture ontology: *A farmer is a person who operates a farm.*

Initially, LExO applies the Minipar dependency parser [21] in order to produce a structured output as shown in Figure 1. Every node in the dependency tree contains information about the token such as its lemma (base form), its syntactic category (e.g. *N* (noun)) and grammatical role (e.g. *subj*), as well as its surface position. Indentation in this notation visualizes direct dependency, i.e. each child node is syntactically dominated by its parent.

This dependency structure is now being transformed into an XML-based format (see Figure 2) in order to facilitate the subsequent transformation process, and to make LExO more independent of the particular parsing component.

The set of rules which are then applied to the XML-based parse tree make use of XPath expressions for transforming the dependency structure into one or more OWL DL axioms. Figure 3 shows a few examples of such transformation rules in original syntax. Each of them consists of several arguments (e.g. *arg_1:...*), the values of which are defined by an optional prefix, i.e. a reference to a previously matched argument (*arg_0*), plus an XPath expression such as */C[@role='rel']* being evaluated relative to that prefix. The last lines of each transformation rule define one or more templates for OWL axioms, with variables to be replaced by the values of the arguments. Complex expressions such as *0-1* allow for "subtracting" individual subtrees from the overall tree structure. A more complete listing of the transformation rules we applied can be found further below.

[2] http://ontoware.org/projects/lexo/

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <C id="E1" pos="0">
    <VBE id="3" pos="3" role="i" phrase="is" base="be">
      <N id="2" pos="2" role="s" phrase="farmer">
        <Det id="1" pos="1" role="det" phrase="A"/>
      </N>
      <N id="5" pos="6" role="pred" phrase="person">
        <N id="E3" pos="4" role="subj" base="farmer" antecedent="2"/>
        <Det id="4" pos="5" role="det" phrase="a"/>
        <C id="E0" pos="7" role="rel">
          <N id="6" pos="8" role="whn" phrase="who" antecedent="5"/>
          <V id="7" pos="9" role="i" phrase="operates" base="operate">
            <N id="E4" pos="10" role="subj" base="who" antecedent="5"/>
            <N id="9" pos="12" role="obj" phrase="farm">
              <Det id="8" pos="11" role="det" phrase="a"/>
</N> </V> </C> </N> </VBE> </C> </root>
```

Figure 2. XML Representation of Dependency Tree

A minimal set of rules for building a complete axiomatization of the *Farmer* example could be, e.g. *Copula*, *Relative Clause* and *Transitive Verb Phrase* (see Table 1). The resulting list of axioms (see Figure 4) in KAON2[3] internal syntax is directly fed into the ontology management system which interprets the textual representation of these axioms, and finally builds an unfolded[4] class description as shown in Figure 5.

```
[equivalent lexo:a_farmer lexo:a_person_who_operates_a_farm]
[equivalent lexo:a_person_who_operates_a_farm [and lexo:a_person lexo:operates_a_farm]]
[equivalent lexo:operates_a_farm [some lexo:operates lexo:a_farm]]
```

Figure 4. Resulting Axioms

```
[equivalent lexo:a_farmer [and lexo:a_person [some lexo:operates lexo:a_farm]]]
```

Figure 5. Class Description (unfolded)

Obviously, all parts of this class description have to be normalized. After the normalization, the final, unfolded axiomatization in DL syntax reads:

$$\textit{Farmer} \equiv \textit{Person} \sqcap \exists \textit{operate.Farm}$$

Additionally, it may be necessary to map the ontology elements of the axiomatization to already existing content of the ontology before the results can be used to generate suggestions for ontology changes (cf. Section 6). As shown by the large body of research done in the domain of ontology mapping, this task is not trivial at all. Semantic ambiguities of labels (e.g. homonymy or polysemy), as well as the fact that a single entity or axiom in the ontology can have arbitrarily many lexicalizations – differing even in their syntactic category – make it necessary to consider a multitude of possible mappings. Moreover, idiomatic expressions, i.e. expressions the meaning of which cannot be directly derived from the meaning of their individual components, need to be treated properly. Therefore, in addition to integrating a state-of-the-art mapping framework, a significant degree of user involvement will be unavoidable in the end (see Section 6).

[3] `http://kaon2.semanticweb.org`

[4] By *unfolding*, a term borrowed from logic programming, we mean transformations like that of $\{A \equiv \exists R.B, C \equiv A \sqcap D\}$ to $\{C \equiv \exists R.B \sqcap D\}$. The specific for of output which we receive allows us to remove many of the newly generated class names by unfolding, in order to obtain a more concise output.

3.2.2. Transformation Rules

Table 1 gives an overview of the most frequently used transformation rules. Each row in the table contains the rule name (e.g. *Verb with Prepositional Complement*) and an expression describing the natural language syntax matched by that rule – like, for example, $V_0\ Prep_0\ NP(\textit{pcomp-n})_0$, where V_0 represents a verb, $Prep_0$ a preposition and $NP(\textit{pcomp-n})$ denotes a noun phrase acting as a prepositional complement. Please note that these expressions are very much simplified for the sake of presentation. The last column shows the OWL axioms generated in each case, where X denotes the atomic class name represented by the surface string of the complete expression matched by the regarding transformation rule.

It is important to emphasize that this set of rules is by no means exhaustive, nor does it define the only possible way to perform the transformation. In fact, there are many different modeling possibilities, and the choice and shape of the rules very much depends on the underlying application, the domain of interest or individual modeling preferences of the user (see example *Tetraploid* in Section 7).

Rule	Natural Language Syntax	OWL Axioms
Disjunction	NP_0 `or` NP_1	$X \equiv \text{NP}_0 \sqcup \text{NP}_1$
Conjunction	NP_0 `and` NP_1	$X \equiv \text{NP}_0 \sqcap \text{NP}_1$
Determiner	$\text{Det}_0\ \text{NP}_0$	$X \equiv \text{NP}_0$
Intersective Adjective	$\text{Adj}_0\ \text{NP}_0$	$X \equiv \text{Adj}_0 \sqcap \text{NP}_0$
Subsective Adjective	$\text{Adj}_0\ \text{NP}_0$	$X \sqsubseteq \text{NP}_0$
Privative Adjective	$\text{Adj}_0\ \text{NP}_0$	$X \sqsubseteq \neg\text{NP}_0$
Copula	NP_0 VBE NP_1	$\text{NP}_0 \equiv \text{NP}_1$
Relative Clause	NP_0 C(*rel*) VP_0	$X \equiv \text{NP}_0 \sqcap \text{VP}_0$
Number Restriction	V_0 Num NP(obj)_0	$X \equiv =\text{Num}\ \text{V}_0.\text{NP}_0$
Negation (not)	`not` $\text{V}_0\ \text{NP}_0$	$X \sqsubseteq \neg\exists \text{V}_0.\text{NP}_0$
Negation (without)	NP_0 `without` $\text{NP}(\textit{pcomp-n})_1$	$X \equiv \text{NP}_0 \sqcap \neg\text{with}.\text{NP}_1$
Participle	$\text{NP}_0\ \text{VP}(\textit{vrel})_0$	$X \equiv \text{NP}_0 \sqcap \text{VP}_0$
Transitive Verb Phrase	$\text{V}_0\ \text{NP}(\textit{obj})_0$	$X \equiv \exists \text{V}_0.\text{NP}_0$
Verb with Prep. Compl.	$\text{V}_0\ \text{Prep}_0\ \text{NP}(\textit{pcomp-n})_0$	$X \equiv \exists \text{V}_0\_\text{Prep}_0.\text{NP}_0$
Noun with Prep. Compl.	$\text{NP}_0\ \text{Prep}_0\ \text{NP}(\textit{pcomp-n})_1$	$X \equiv \text{NP}_0 \sqcap \exists \text{NP}_0\_\text{Prep}_0.\text{NP}_1$
...	...	...

Table 1. Transformation Rules

3.3. Learning Disjointness

The feasibility of learning disjointness based on simple lexical evidence in principle has already been shown in [22]. However, our experiments indicate that a single heuristic is not suitable for detecting disjointness with sufficiently high precision, i.e. better than an average human could do.

An extensive survey which we performed in order to collect experience with modeling disjoint classes revealed several problems frequently encountered by users who try to introduce disjointness axioms. Based on the results of this survey we developed a variety of different methods in order to automatically extract lexical *and* logical features which we believe to provide a solid basis for learning disjointness [27]. These methods take

into account the structure of the ontology, associated textual resources, and other types of data sources in order to compute the likeliness of two classes to be disjoint. The features obtained from these methods are used to train a classifier that decides whether any given pair of classes is disjoint or not. In the remainder of this Section, we will describe those features in more detail before concluding with a summary of our experiments and evaluation results.

3.3.1. Taxonomic Overlap

In description logics, two classes are disjoint *iff* their "taxonomic overlap"[5] *must* be empty. Because of the open world assumption in OWL, the individuals of a class do not necessarily have to *exist* in the ontology. Hence, the taxonomic overlap of two classes is considered not empty as long as there *could* be common individuals within the domain of interest which is modeled by the ontology, i.e. if the addition of such an individual does *not* generate an inconsistency.

We developed three methods which determine the likeliness for two classes to be disjoint by considering their overlap with respect to (i) *individuals* and *subclasses* in the ontology – or learned from a corpus of associated textual resources – and (ii) *Del.icio.us*[6] documents tagged with the corresponding class labels. An additional feature indicating disjointness is computed by determining whether (iii) any of the classes is *subsumed* by the other.

Ontology Individuals and subclasses which may serve as indicators for taxonomic overlap can be imported either from an ontology, or from a given corpus of text documents. In the latter case, `subclass-of` and `instance-of` relationships are extracted by different algorithms provided by the Text2Onto[7] ontology learning framework. A detailed description of these algorithms can be found in [23]. All taxonomic relationships – learned and imported ones – are associated with rating annotations $r_{\text{subclass-of}}$ (or $r_{\text{instance-of}}$ respectively) indicating the certainty of the underlying ontology learning framework in the correctness of its results. For imported relationships the confidence is 1.0.

The following feature $\mathrm{f}_{subclass-of}$ models the confidence for a pair (c_1, c_2) to be *not* disjoint based on the taxonomic overlap of c_1 and c_2 with respect to common subclasses (and in a similar way for instances):

$$\mathrm{f}_{subclass-of}(c_1, c_2) = \frac{\sum_{c \sqsubseteq c_1 \sqcap c_2} (r_{\text{subclass-of}}(c, c_1) \cdot r_{\text{subclass-of}}(c, c_2))}{\sum_{c \sqsubseteq c_1} r_{\text{subclass-of}}(c, c_1) + \sum_{c \sqsubseteq c_2} r_{\text{subclass-of}}(c, c_2)} \quad (1)$$

Del.icio.us Del.icio.us is a server-based system with a simple-to-use interface that allows users to organize and share bookmarks on the internet. It associates each URL with a description, a note, and a set of tags (i.e. arbitrary class labels). For our experiments, we collected $|U| = 75,242$ users, $|T| = 533,191$ tags and $|R| = 3,158,297$ resources, related by $|Y| = 17,362,212$ triples. The idea underlying the use of del.icio.us in this

[5] We use this notion to refer to the set of common individuals.

[6] `http://del.icio.us`

[7] `http://ontoware.org/projects/text2onto/`

case is that two labels which are frequently used to tag the same resource are likely to be disjoint, because users tend to avoid redundant labeling of documents.

In this case, we compute the confidence that c_1, c_2 are *not* disjoint as

$$\mathrm{f}_{del.icio.us}(c_1, c_2) = \frac{|\{d : c_1 \in t(d), c_2 \in t(d)\}|}{\sum_{c \in C} |\{d : c_1 \in t(d), c \in t(d)\}| + \sum_{c \in C} |\{d : c_2 \in t(d), c \in t(d)\}|} \tag{2}$$

where C is the set of all classes and $t(d)$ represents the set of del.icio.us *tags* associated with document d. The normalized number of co-occurrences of c_1 and c_2 (their respective labels to be precise) as del.icio.us tags aims at capturing the degree of association between the two classes.

Subsumption A particular case of taxonomic overlap is subsumption, which provides us with additional evidence with respect to the disjointness of two classes. If one class is a subclass of the other we assume these two classes to be *not* disjoint with a confidence equal to the likeliness $r_{\text{subclass-of}}$ associated with the `subclass-of` relationship.

3.3.2. Semantic Similarity

The assumption that a direct correspondence between the semantic similarity of two classes and their likeliness to be disjoint led to the development of three further methods: The first one implements the similarity measure described by [24] to compute the semantic similarity *sim* of two classes c_1 and c_2 with respect to *WordNet* [25]:

$$\mathrm{f}_{wordnet}(c_1, c_2) = \mathrm{sim}(s_1, s_2) = \frac{2 * \mathrm{depth}(\mathrm{lcs}(s_1, s_2))}{\mathrm{depth}(s_1) + \mathrm{depth}(s_2)} \tag{3}$$

where s_i denotes the first sense of c_i, $i \in \{1, 2\}$ with respect to WordNet, and $\mathrm{lcs}(s_1, s_2)$ is the least common subsumer of s_1 and s_2. The depth of a node n in WordNet is recursively defined as follows: $\mathrm{depth}(\mathrm{root}) = 1$, $\mathrm{depth}(\mathrm{child}(n)) = \mathrm{depth}(n) + 1$.

The second method measures the distance of c_1 and c_2 with respect to the given background *ontology* by computing the minimum length of a path of `subclass-of` relationships connecting c_1 and c_2.

$$\mathrm{f}_{ontology}(c_1, c_2) = \min_{p \in \mathrm{paths}(c_1, c_2)} \mathrm{length}(p) \tag{4}$$

And finally, the third method computes the similarity of c_1 and c_2 based on their *lexical context*. Along with the ideas described in [17] we exploit Harris' distributional hypothesis [26] which claims that two words are semantically similar to the extent to which they share syntactic contexts.

For each occurrence of a class label in a corpus of textual documents (see preliminaries of this section) we consider all the lemmatized tokens in the same sentence (except for stop words) as potential features in the context vector of the corresponding class. After the context vectors for both classes have been constructed, we assign weights to all

features by using a modified version of the TFIDF formula. It differs from the original version in that it aims at measuring the significance of terms with respect to the classes they co-occur with rather than the documents in which they are contained.

Let $v_c = (f_{c,1}, ...f_{c,n})$, $n \geq 1$ be the context vector of class c where each $f_{c,j}$ is the frequency of token t_j in the context of c. Then we define $TF_{c,j} = f_{c,j} \cdot (\sum_{1 \leq k \leq n} f_{c,k})^{-1}$ and $DF_j = |\sum_{c' \in C} f_{c',j} > 0|$, where C is the set of all classes. Finally, we get $f'_{c,j} = TF_{c,j} \cdot \log(|C| \cdot (DF_j)^{-1})$, hence $v'_c = (f'_{c,1}, ...f'_{c,n})$. Given the *weighted* context vectors v'_{c_1} and v'_{c_2} the confidence in c_1 and c_2 being *not* disjoint is defined as $f_{context}(c_1, c_2) = v'_{c_1} \cdot v'_{c_2} \cdot (\|v'_{c_1}\| \|v'_{c_2}\|)^{-1}$ which corresponds to the cosine similarity of v'_{c_1} and v'_{c_2}.

3.3.3. Patterns

Since we found that disjointness of two classes is often reflected by human language, we defined a number of lexico-syntactic patterns to obtain evidence for disjointness relationships from a given corpus of textual resources. The first type of pattern is based on enumerations as described in [22]. The underlying assumption is similar to the idea described in section 3.3.1, i.e. terms which are listed separately in an enumeration mostly denote disjoint classes. Therefore, from the sentence

The pigs, cows, horses, ducks, hens and dogs all assemble in the big barn, thinking that they are going to be told about a dream that Old Major had the previous night.[8]

we would conclude that *pig*, *cow*, *horse*, *duck*, *hen* and *dog* denote disjoint classes. This is because we believe that – except for some idiomatic expressions it would be rather unusual to enumerate overlapping classes such as *dogs* and *sheep dogs* separately which would result in semantic redundancy. More formally:

Given an enumeration of noun phrases $NP_1,\ NP_2,\ \ldots,\ (and|or)\ NP_n$ we conclude that the concepts $c_1,\ c_2,\ \ldots,\ c_k$ denoted by these noun phrases are pairwise disjoint, where the confidence $\mathrm{f}_{enumeration}(c_1, c_2)$ for the disjointness of two concepts c_1 and c_2 is obtained from the number of evidences found for their disjointness in relation to the total number of evidences for the disjointness of these concepts with other concepts.

The second type of pattern is designed to capture more explicit expressions of disjointness in natural language by phrases such as $either\ NP_1\ or\ NP_2$ or $neither\ NP_1\ nor\ NP_2$. For both types of patterns we compute the confidence for the disjointness of two classes c_1 and c_2 as follows:

$$\mathrm{f}_{pattern}(c_1, c_2) = \frac{\mathrm{freq}(c_1, c_2)}{\sum_{j \neq 1} \mathrm{freq}(c_1, c_j) + \sum_{i \neq 2} \mathrm{freq}(c_i, c_2)} \tag{5}$$

where $freq(c_i, c_j)$ is the number of patterns providing evidence for the disjointness of c_i and c_j with $0 \leq i, j \leq |C|$ and $i \neq j$.

3.3.4. Evaluation

We evaluated the approach by performing a comparison of learned disjointness axioms with a data set consisting of 2,000 pairs of classes, each of them manually tagged by

[8] George Orwell, *Animal Farm*, Secker & Warburg, London, 1945

Table 2. Evaluation against Majority Vote 100% (ADTree)

Dataset	P			R			F			Acc	$Acc_{majority}$
	+	−	avg.	+	−	avg.	+	−	avg.		
$Experts$	0.896	0.720	0.808	0.903	0.703	0.803	0.899	0.712	0.806	0.851	0.738
$Students$	0.866	0.790	0.828	0.942	0.599	0.771	0.903	0.681	0.792	0.851	0.734
$Avg.$	0.881	0.755	0.818	0.923	0.651	0.787	0.901	0.697	0.799	0.851	0.736
All	0.934	0.823	0.879	0.946	0.789	0.868	0.940	0.805	0.873	0.909	0.760

6 human annotators – 3 students and 3 ontology experts[9]. A 10-fold cross validation against those pairs of classes which were tagged identically by all the annotators showed an accuracy between 85.1% and 90.9%, which is significantly higher than the majority baseline (cf. Table 2).

In order to find out which classification features contributed most to the overall performance of the classifier we performed an analysis of our initial feature set with respect to the gain ratio measure. The ranking produced for data set C clearly indicates an exceptionally good performance of the features taxonomic overlap (Section 3.3.1), similarity based on WordNet and lexical context (Section 3.3.2), and del.icio.us (Section 3.3.1). The contribution of other features such as the one presented in Section 3.3.3 relying on lexico-syntactic patterns seems to be less substantial. However, as the classification accuracy tested on every single feature is always below the overall performance, the combination of all features is necessary to achieve a very good overall result.

4. Discussion

The syntactic transformation proposed in Section 3.2 creates a set of OWL axioms which can be used to extend the axiomatization of any given class in an ontology. Our naive implementation of this approach is as simple as efficient, but obviously requires a significant amount of manual or automatic post-processing. This is to a major extent due to a number of problems which relate to limitations of the linguistic analysis and the transformation process, as well as fundamental differences between lexical and ontological semantics. In the following we will discuss some of these problems in more detail, and present possible solutions.

Although the transformation takes into account some aspects of lexical semantics, it is certainly not capable of capturing much of the intension of the terms involved in the natural language expression that serves as an input for the transformation process. Much of the meaning of the resulting axioms is still brought in by the semantics of the underlying natural language terms. This does not necessarily constitute a significant problem as long as the semantics of the description logic expressions is sufficiently "in line" with the lexical semantics of the terms involved in their formation. Actually, the semantics of ontological elements – not of the constructs of the ontology language, but of the classes, properties and instances defined by means of these constructs – will always be grounded to some extent in natural language semantics.

[9]The complete data set is available from `http://ontoware.org/projects/leda/`.

As it is impossible to express all possible aspects of a concept's meaning by virtue of description logic axioms, natural language labels and comments undoubtedly play a key role in ontological knowledge representation. In fact, an ontology without natural language labels attached to classes or properties is almost useless, because without this kind of grounding it is very difficult, if not impossible, for humans to map an ontology to their own conceptualization, i.e. the ontology lacks human-interpretability.

However, a grounding of ontologies in natural language is highly problematic due to different semantics and the dynamic nature of natural language. It is important to mention that many problems linked to either of these aspects are not necessarily specific to ontology learning approaches such as the one we present in this chapter. Since the way people conceive and describe the world is very much influenced by the way they speak and vice-versa (also known as the *Sapir-Whorf hypothesis*), ontology engineering is often subject to our intuitive understanding of natural language semantics. Some problems that relate to differences between ontological and lexical semantics are discussed in the following.

Lexical Semantics. The semantics of lexical relations fundamentally differs from the model-theoretic semantics of ontologies. While lexical relations such as hyponymy, antonymy or synonymy are defined over lexemes, ontological relations are used for relating classes.[10] And it is not obvious in all cases how to map words – especially very abstract notions – to classes, as their extension often remains unclear.

For practical reasons it might be sensible to assume a correspondence between lexical relations and some types of axioms. Indeed, traditional ontology learning approaches often rely on information about hyponymy and synonymy for creating subsumption hierarchies [12], or meronymy for identifying part-of relationships [28]. However, a one-to-one mapping between lexical and ontological semantics is problematic for various reasons. Just to mention a few of them, true synonymy is very rare if existent at all in natural language. And since tiny differences in meaning may be significant depending on the modeling granularity of an ontology, synonymy cannot always be mapped to equivalence in a straightforward way. Second, lexical relations such as meronymy do not need to be transitive even if their ontological counterparts are. It is also important to mention that hyponymy in pattern-based ontology learning is often confused with *para-hyponymy* [29] as those patterns are not able to capture the necessity condition which holds for regular hyponymy[11]. Finally, one has to be aware of the fact that such a mapping between lexical and model-theoretic semantics may affect the formal correctness of an ontology – even more, if ontology learning or engineering exclusively relies on lexico-syntactic clues for inferring lexical relationships. Due to the informal character of natural language it is no trouble to say, for instance, "*A person is an amount of matter*". But from the perspective of formal semantics this might be problematic as pointed out by [30].

[10] For example, each of these classes could be associated with one or more natural language expressions describing the intended meaning (intension) of the class. And still, since hyponymy is not "transitive" over (near-)synonymy it is not necessarily the case that all mutually synonymous words associated with a subclass are hyponyms of all synonymous words associated with its superclass.

[11] Interestingly, this necessity condition parallels the *rigidity* constraint as defined by the OntoClean methodology [30]. A tool such as AEON [36] could therefore help to automatically detect both, cases of formally incorrect subsumption as well as para-hyponymy relationships (e.g. "A *dog* is a *pet*.").

Dynamics of Natural Language. Further problems with respect to the use of natural language in ontology engineering relate to the way in which semantics are defined. While ontologies have a clear model-theoretic semantics, the semantics of lexical relations is defined by so-called *diagnostic frames*, i.e. by typical sentences describing the context in which a pair of words may or may not occur given a certain lexical relation among them. This way of defining lexical relations does not guarantee for stable semantics, since natural languages, other than ontology representation languages, are dynamic. That means, each (*open-class*) word slightly changes its meaning every time it is used in a new linguistic context. These *semantic shifts*, if big enough, can affect the lexical relationships between any pair of words. And considering that natural language expressions are regularly used for the grounding of ontologies they can potentially lead to semantic "inconsistencies", i.e. conflicting intensional descriptions. This kind of inconsistencies can be avoided by more precise, formal axiomatizations of ontological elements. However, it is an open issue how many axioms are required to "pin down" the meaning of a given class or property.

The proposed approach for learning disjointness axioms (see Section 3.3) is affected by similar problems as it relies among others upon ontology learning methods for capturing the potential overlap of any two concepts. However, one of the main weaknesses of this approach might be that it crucially depends on the quality of the manually created training data sets. Our user study [27] revealed a number of difficulties and misunderstandings human ontology engineers had while creating disjointness axioms. For example, a simple taxonomy along with natural language labels was often not sufficient for disambiguating the sense of a given concept. And people were confused if the intensions of two concepts were disjoint while their extensions were not – or vice versa (e.g. *Woman* and *US President*).

5. Dealing with Inconsistencies in Ontology Learning

One of the major problems of learning ontologies is the potential introduction of inconsistencies. These inconsistencies are a consequence of the fact that it is inherent in the ontology learning process that the acquired ontologies represent *imperfect* information.

According to [31], we can distinguish three different causes of imperfection. Imperfection can be due to *imprecision*, *inconsistency* or *uncertainty*. Imprecision and inconsistency are properties of the information itself – either more than one world (in the case of ambiguous, vague or approximate information) or no world (if contradictory conclusions can be derived from the information) is compatible with the given information. Uncertainty means that an agent, i.e. a computer or a human, has only partial knowledge about the truth of a given piece of information. One can distinguish between objective and subjective uncertainty. Whereas objective uncertainty relates to randomness referring to the propensity or disposition of something to be true, subjective uncertainty depends on an agent's opinion about the truth of some information. In particular, an agent can consider information as unreliable or irrelevant.

In ontology learning, (subjective) uncertainty is the most prominent form of imperfection. This is due to the fact that the results of the different algorithms have to be considered as unreliable or irrelevant due to imprecision and errors introduced during the ontology generation process. There exist different approaches for the representation of

uncertainty: Uncertainty can for example be represented as part of the learned ontologies, e.g. using probabilistic extensions to the target knowledge representation formalism, or at a meta-level as application-specific information associated with the learned structures.

Ignoring the fact that learned ontologies contain uncertain and thus potentially contradicting information would result in highly inconsistent ontologies, which do not allow meaningful reasoning. In the following we show how inconsistencies can be dealt with in the process of ontology learning. In particular, we show how the concept of consistent ontology evolution can be applied in the context of ontology learning. To begin with, we define the notion of consistency more precisely.

5.1. Logical Consistency

Logical consistency addresses the question whether the ontology is "semantically correct", i.e. does not contain contradicting information. We say *logical consistency* is satisfied for an ontology O if O is satisfiable, i.e. if O has a model. Please note that because of the monotonicity of the considered logic, an ontology can only become logically inconsistent by adding axioms: If a set of axioms is satisfiable, it will still be satisfiable when any axiom is deleted. Therefore, we only need to check the consistency for ontology change operations that add axioms to the ontology. Effectively, if $O \cup \{\alpha\}$ is inconsistent, in order to keep the resulting ontology consistent some of the axioms in the ontology O have to be removed or altered.

Example. Suppose, we have generated an ontology containing the following axioms: Pig $\sqsubseteq$ Mammal, Human $\sqsubseteq$ Mammal, Human $\sqsubseteq$ Biped (humans walk on two legs), Pig $\sqsubseteq$ Quadruped (pigs walk on two legs), Biped $\sqsubseteq \neg$Quadruped (Bipeds and Quadrupeds are disjoint), Pig(OldMajor). This ontology is logically consistent.

Suppose we now learn from some source that Old Major walks on two legs and want to add the axiom Biped(OldMajor). Obviously, this ontology change operation would result in an inconsistent ontology.

5.2. Consistent Ontology Evolution

As we have already discussed in Section 2.2, the most adequate approach to dealing with inconsistencies in ontology learning is by realizing a consistent evolution of the ontology. The goal of consistent ontology evolution is the resolution of a given ontology change in a systematic manner by ensuring the consistency of the whole ontology. It is realized in two steps:

1. *Inconsistency Localization*: This step is responsible for checking the consistency of an ontology with the respect to the ontology consistency definition. Its goal is to find "parts" in the ontology that do not meet consistency conditions;
2. *Change Generation*: This step is responsible for ensuring the consistency of the ontology by generating additional changes that resolve detected inconsistencies.

The first step essentially is a diagnosis process. There are different approaches how to perform the diagnosis step [32]. A typical way to diagnose an inconsistent ontology is to try to find a minimal inconsistent subontology, i.e. a minimal set of contradicting axioms. Formally, we call an ontology O' a *minimal inconsistent subontology* of O,

if $O' \subseteq O$ and O' is inconsistent and for all O'' with $O'' \subset O'$, O'' is consistent. Intuitively, this definition states that the removal of any axiom from O' will result in a consistent ontology. A simple way of finding a minimal inconsistent subontology is as follows: We start with one candidate ontology containing initially only the axiom that was added to the ontology as part of the change operation. As long as we have not found an inconsistent subontology, we create new candidate ontologies by adding axioms (one at a time) that are in some way connected with the axioms in the candidate ontology. One simple, but useful notion of connectedness is structural connectedness: We say that axioms are structurally connected if they refer to shared ontology entities. Once the minimal inconsistent ontology is found, it is by definition sufficient to remove any of the axioms to resolve the inconsistency.

In our previous example, a minimal inconsistent subontology would consist of the axioms Pig $\sqsubseteq$ Quadruped, Biped $\sqsubseteq$ $\neg$Quadruped, Pig(OldMajor), and Biped(OldMajor). The removal of any of the axioms would result in a consistent ontology.

While the removal of any of the axioms from a minimal inconsistent subontology will resolve the inconsistency, the important question of course is deciding *which* axiom to remove. This problem of only removing dispensable axioms requires some semantic selection functions capturing the relevance of particular axioms. These semantic selection functions can for example exploit information about the confidence in the axioms that allows us to remove "less correct" axioms. In the resolution of the changes we may decide to remove the axioms that have the lowest confidence, i.e. those axioms that are most likely incorrect. We are thus able to incrementally evolve an ontology that is (1) consistent and (2) captures the information with the highest confidence. For details of such a process and evaluation results, we refer the reader to [22].

Based on the discussions above, we can now outline an algorithm (c.f. Algorithm 1) to ensure the consistent evolution of a learned ontology.

Algorithm 1 Algorithm for consistent ontology learning

Require: A consistent ontology O
Require: A set of ontology changes OC
for all $\alpha \in OC, r_{\text{conf}}(\alpha) \geq t$ **do**
  $O := O \cup \{\alpha\}$
  while O is inconsistent **do**
    $O' :=$ minimal_inconsistent_subontology(O, α)
    $\alpha^- := \alpha$
    for all $\alpha' \in O'$ **do**
      if $r_{\text{conf}}(\alpha') \leq r_{\text{conf}}(\alpha)$ **then**
        $\alpha^- := \alpha'$
      end if
    end for
    $O := O \setminus \{\alpha^-\}$
  end while
end for

Starting with some consistent ontology O, we incrementally add all axioms generated from the ontology learning process – contained in the set of ontology changes OC

– whose confidence is equal to or greater than a given threshold t. If adding the axioms leads to an inconsistent ontology, we localize the inconsistency by identifying a minimal inconsistent subontology. Within this minimal inconsistent subontology we then identify the axiom that is most uncertain, i.e. has the lowest confidence value. This axiom will be removed from the ontology, thus resolving the inconsistency. It may be possible that one added axiom introduced multiple inconsistencies. For this case, the above inconsistency resolution has to be applied iteratively.

5.3. Context Information for the Resolution of Inconsistencies

Besides the general notion of confidence used above, we may rely on various other forms of contextual information to obtain a ranking of the axioms for the resolution of inconsistencies. In the following we discuss what kind of contextual information can be automatically generated by the ontology learning algorithms:

Axiomatic support. The axiomatic support can be defined as the relative number of times a particular axiom was generated by an ontology learning component. Since the methods which are applied by such a component can generate axioms of different complexity, it may be necessary to define the axiomatic support based on some normal form of the axioms.

Provenance information. Whenever ontologies are automatically generated from structured or unstructured resources, and in particular if these resources are part of the World Wide Web, the reliability of the results depends on the trustworthiness and quality of these resources. Therefore, associating provenance information with the learned ontology elements does not only increase the traceability of the results (as the user can track individual elements back to the resources they have been extracted from), but also helps to estimate the correctness of the results.

Mapping confidence. If the ontology learning task is to extend a given ontology (as opposed to generating an ontology from scratch) it can be necessary to map all newly introduced properties and class descriptions to already existing ontology elements. This helps to avoid unnecessary extensions to the ontology, but at the same time introduces additional uncertainty caused by incorrect mappings, as suggested in Section 3.2.

Rule reliability. Ontology learning approaches based on syntactic transformation rules (c.f. Section 3.2.2), or lexico-syntactic patterns [18] often make certain assumptions about the reliability of their rules or patterns. These (implicit or explicit) assumptions typically being supported by empirical data can be used to estimate the correctness of the ontology learning results.

Classifier confidence. Supervised ontology learning approaches such as the one presented in Section 3.3 rely on a classification model built from training examples. The classifier that can be constructed from this model will make predictions for previously unseen data (e.g. instances to be classified as belonging to a certain class, or pairs of classes being disjoint or not) with a confidence value that depends on the classifier type.

Relevance. In general, ontology learning from text is based on the assumption that the domain of interest, i.e. the domain to be modeled by the learned ontology, is given by means of the underlying document corpus. It therefore seems natural that approaches

such as [34] try to evaluate the quality, and in particular the domain coverage, of learned ontologies by comparing them to the corpus. Similarly, the relevance of an individual ontology element can be estimated based on the pieces of evidence (e.g. explicit mentions) in the corpus.

6. Integrating Learning and Evolution into the Ontology Lifecycle

In this section we sketch our vision of a semi-automatic ontology engineering process, which integrated our previously described methods for ontology learning and evolution along with an elaborate methodology. We describe the potential role of our approaches within this scenario and identify the missing components.

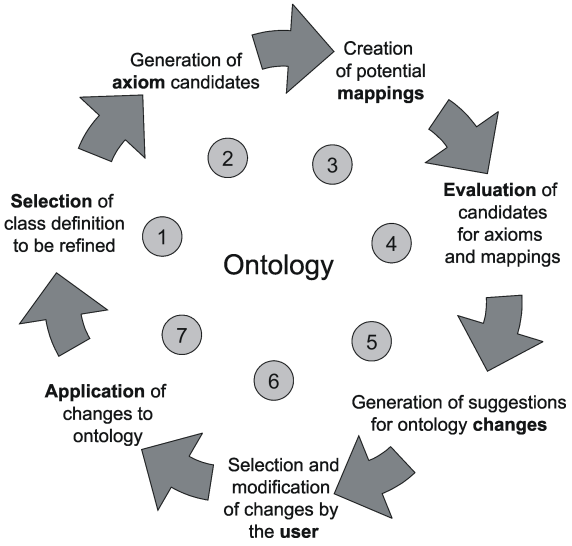

Figure 6. Ontology Evolution Process

The overall scenario we envision for the evolution of expressive OWL ontologies is a semi-automatic cyclic process of ontology learning, evaluation and refinement as depicted by Figure 6. The process starts with a relatively inexpressive ontology, possibly a bare taxonomy, which is supposed to be enriched and refined to meet the requirements, e.g. of a reasoning-based application. In each iteration of the process, the user selects the class to be refined, and optionally specifies appropriate resources for the ontology generation phase (Step 1) such as

- manual user input,
- comments contained in the ontology,
- definitions extracted from ontology engineering discussions by email or Wiki,
- software documentation of the underlying application,
- available glossaries and encyclopedias (e.g. Wikipedia), or
- textual descriptions of the domain which could be obtained by initiating a Google™search for definitions (e.g. "*define: DNS*").

A tool such as LExO (cf. Section 3.2) can analyze the given resources to identify and extract definitory sentences, i.e. natural language descriptions of the class previously selected by the user. These definitions are parsed and transformed into OWL DL axioms (Step 2) that can be presented to the user, if she wants to intervene at this point.

Otherwise, the system directly proceeds to the mapping phase which aims at relating the newly generated entities and axioms to elements in the initial ontology (Step 3). The outcome of this phase are a number of mapping axioms which can be added to the class axiomatization after being confirmed by the user. Then, methods for consistent ontology evolution check for logical inconsistencies or potential modeling errors (Step 4). Based on the learned axiomatization and additional mappings the system now suggests ontology changes or extensions to the user (Step 5). The user now revises the ontology by modifying or removing some of the axioms (Steps 6 and 7), before the whole process starts over again. Further entities, e.g. those introduced by previous iterations, can be refined until the user or application needs are satisfied.

When validating the ontology, it is certainly necessary to consider aspects beyond that of logical consistency. We point out two aspects which we judge to be of particular importance, namely how to aid the ontology engineer to ensure on the one hand a sufficiently high *quality* of the ontology and on the other hand the *completeness* of the modeling process in terms of the application domain.

Quality insurance will have to be based on previous work on the field of ontology evaluation. Since the automatic generation of expressive ontologies can potentially lead to a substantial increase in complexity, a simple manual revision of the ontology generated by a system such as the one described here might be infeasible. Therefore, we believe that automatic techniques for ontology evaluation will play a crucial role in the ontology learning and engineering cycle. These techniques could check, for instance, the ontology's coverage with respect to a domain-specific corpus [34] or its validity in terms of the OntoClean methodology [36].

In order to ensure completeness of the modeling process with regard to the application domain, a structured approach for an exhaustive exploration of complex relationships between classes is required. This can be realized, for example, by employing methods like *relational exploration* [37], which is an adaptation of attribute exploration from Formal Concept Analysis [38] to description logics. And finally, it might also be worthwhile to consider an integration of LExO with other learning approaches which could compensate for some of its limitations, e.g. with respect to the learnability of particular relations between roles [39], or disjointness axioms (see Section 3.3).

7. Experiments in an Application Scenario

In this section we discuss an application scenario of ontology learning in the context of a case study in the fishery's domain at the Food and Agriculture Organization (FAO) of the UN.

The FAO Fisheries department has several information and knowledge organization systems to facilitate and secure the long-term, sustainable development and utilization of the world's fisheries and aquaculture. In order to effectively manage the world's shared fish stocks and prevent overfishing, the FAO Fishery systems manage and disseminate statistical data on fishing, GIS data, information on aquaculture, geographic entities, de-

scription of fish stocks, etc. However, even though much of the data is "structured", it is not necessarily represented in a formal way, and some of the information resources are not available through databases but only as parts of websites, or as individual documents or images. Therefore, many or even all of these data sources could be better exploited by bringing together related and relevant information, along with the use of the fishery ontologies, to provide inference-based services for policy makers and national governments to make informed decisions.

A particular application developed within the NeOn project[12] is FSDAS (Fishery Stock Depletion Alert System), an ontology-driven decision support system for fisheries managers, assistants to policy makers and researchers. FSDAS will be a web-based intelligent agent that uses networked ontologies consisting of various fisheries, taxonomic, and geographical ontologies to aid users in discovering resources and relationships related to stock depletion and to detect probabilities of over-fishing. Fisheries ontologies, which bring together concepts from a number of existing knowledge organization systems, help to improve language-independent extraction and the discovery of information. Their development will allow for managing the complexity of fishery knowledge communities, their meaning negotiation and their deployment by worldwide authorities.

In order to achieve these goals, the ontological model needs to be shaped starting from highly structured FAO information systems, and to develop a learning capacity from this model to incorporate data and information from other less structured systems. Here, ontology learning becomes an integral part of the lifecycle of the fishery ontology. Further, in order for the FSDAS to be effective, it is important that the ontologies and resources it builds on are maintained and kept up-to-date, and that when applying changes to ontologies the consistency of the ontology is guaranteed.

7.1. Learning an Ontology for the Fishery Domain

We exemplify our approach by giving a number of axiomatizations automatically generated by means of LExO and the set of rules listed by Table 1. The example sentences are not artificial, but were selected from a fishery glossary provided by FAO.

1. ***Data:*** *Facts that result from measurements or observations.*
   $\text{Data} \equiv \text{Fact} \sqcap \exists \text{result\_from.}(\text{Measurement} \sqcup \text{Observation})$
2. ***InternalRateOfReturn:*** *A financial or economic indicator of the net benefits expected from a project or enterprise, expressed as a percentage.*
   $\text{InternalRateOfReturn} \equiv (\text{Financial} \sqcup \text{Economic}) \sqcap \text{indicator} \sqcap \exists \text{indicator\_of.}(\text{Net} \sqcap \text{Benefit} \sqcap \exists \text{expected\_from.}(\text{Project} \sqcup \text{Enterprise})) \sqcap \exists \text{expressed\_as.Percentage}$
3. ***Vector:*** *An organism which carries or transmits a pathogen.*
   $\text{Vector} \equiv \text{Organism} \sqcap (\text{carry} \sqcup \exists \text{transmit.Pathogen})$
4. ***Juvenile:*** *A young fish or animal that has not reached sexual maturity.*
   $\text{Juvenile} \equiv \text{Young} \sqcap (\text{Fish} \sqcup \text{Animal}) \sqcap \neg \exists \text{reached.}(\text{Sexual} \sqcap \text{Maturity})$
5. ***Tetraploid:*** *Cell or organism having four sets of chromosomes.*
   $\text{Tetraploid} \equiv (\text{Cell} \sqcup \text{Organism}) \sqcap =4 \text{ having.}(\text{Set} \sqcap \exists \text{set\_of.Chromosomes})$
6. ***Pair Trawling:*** *Bottom or mid-water trawling by two vessels towing the same net.*
   $\text{PairTrawling} \equiv (\text{Bottom} \sqcup \text{MidWater}) \sqcap \text{Trawling} \sqcap =2 \text{ trawling\_by.}(\text{Vessel} \sqcap \exists \text{tow.}(\text{Same} \sqcap \text{Net}))$

[12] http://www.neon-project.org

7. ***Sustained Use:*** *Continuing use without severe or permanent deterioration in the resources.*
   SustainedUse $\equiv$ Continuing $\sqcap$ Use $\sqcap$ $\neg\exists$use_with.((Severe $\sqcup$ Permanent) $\sqcap$ Deterioration $\sqcap$ $\exists$deterioration_in.Resources)
8. ***Biosphere****: The portion of Earth and its atmosphere that can support life.*
   Biosphere $\equiv$ Portion $\sqcap$ $\exists$portion_of.((Earth $\sqcap$ Its $\sqcap$ Atmosphere) $\sqcap$ $\exists$can_support.Life)

Some critical remarks and observations on the examples:

1. This is a simple example, which works out very well.
2. This example shows the complex axiomatizations which can be obtained using our approach. Here (and in other examples) we note that adjectives are so far interpreted as being intersective – we discuss this in Section 4. Another recurring problem is the generic nature of the role *of* which we tried to solve by designing the transformation rule in a way that it adds a disambiguating prefix to the preposition as a role name (*indicator_of*). Nevertheless, the output is a reasonable approximation of the intended meaning and would serve well as suggestion for an ontology engineer within an interactive process as we draft in Section 6.
3. This is a Minipar parse error. The desired solution would be
   Vector $\equiv$ Organism $\sqcap$ ($\exists$carry.Pathogen $\sqcup$ transmit.Pathogen).
4. Take particular attention to the handling of negation and of the present perfect tense.
5. The natural language sentence is actually ambiguous whether the number should be read as *exactly four* or *at least four*, and the role name *having* is certainly not satisfactory. Even more difficult is how *set of chromosomes* is resolved. A correct treatment is rather intricate, even if modeling is done manually. The class name *Chromosomes* should probably rather be a nominal containing the class name as individual – which cannot be modeled in OWL DL, but only in OWL Full. Note also that the cardinality restriction is used as a so-called *qualified* one, which is not allowed in OWL DL but is supported by most DL reasoners.
6. *Same* is difficult to resolve. In order to properly model this sentence, one would have to state that two different individuals of the class *Vessel* are connected to the same instantiation of *Net* by means of the *tow* role. This is not expressible in OWL DL as in the general case, such constructions would lead to undecidability.
7. Apart from the very generic role *in* and the problem with adjectives already mentioned, this is a complex example which works very well.
8. The possessive pronoun *its* would have to be resolved.

8. Conclusions

In this chapter we presented two conceptual approaches and implementations for learning expressive ontologies. LExO (cf. Section 3.2) constitutes a lexical approach to generating complex class descriptions from definitory sentences. It can be complemented by any general purpose ontology learning framework, or more specific solutions such as the approach presented in Section 3.3 aiming at the automatic generation of disjointness axioms.

Although we see a great potential in learning expressive ontologies, the discussion shows that there are still many open issues – technical, but also very fundamental ques-

tions. The most important ones according to our perception relate to the relationship of lexical and ontological semantics. Given a purely syntactical transformation such as the one presented in Section 3.2, it will be crucial to investigate at which stage of the process and in which manner particularities of both semantics have to be considered. Finally, we will have to answer the question where the principal limitations of our approaches with respect to the expressivity of the learned ontologies really are. It is reasonable to assume that at least some aspects of ontological semantics cannot (or not so easily) be captured by purely lexical ontology learning methods. However, we believe that a combination of lexical and logical approaches could help to overcome these limitations.

In any case, learning expressive ontologies for knowledge-intensive applications will demand a tighter integration of learning and reasoning at both development time and runtime. One of the most important questions is how potential inconsistencies in the ontology can be dealt with by consistent ontology evolution, for example (see Section 5). Finally, suitable methodologies for semi-automatic ontology engineering will be needed in order to combine ontology learning, evaluation and reasoning.

Acknowledgements

This research has been partially supported by the European Commission under contracts IST-2003-506826 SEKT, IST-2006-027595 NeOn, by the German Federal Ministry of Education and Research (BMBF) under the SmartWeb project 01IMD01B, and by the Deutsche Forschungsgemeinschaft (DFG) under the ReaSem project.

References

[1] J. Hendler, *On beyond ontology*, in: Keynote talk, Second International Semantic Web Conference, 2003.

[2] I. Horrocks and P. F. Patel-Schneider, *Reducing OWL Entailment to Description Logic Satisfiability*, in: Journal of Web Semantics, 1(4):345–357, 2004.

[3] P. Haase, F. van Harmelen, Z. Huang, H. Stuckenschmidt, and Y. Sure, *A framework for handling inconsistency in changing ontologies*, in: Proceedings of the 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, November 6-10, 2005, pages 353–367.

[4] P. Cimiano, J. Völker and R. Studer, *Ontologies on demand? : A description of the state-of-the-art, applications, challenges and trends for ontology learning from text*, in: Information Wissenschaft und Praxis, vol. 57, no. 6–7, pp. 315–320, 2006.

[5] K. Frantzi and S. Ananiadou, *The C-value / NC-value domain independent method for multi-word term extraction*, in: Journal of Natural Language Processing, 6(3), pp. 145–179, 1999.

[6] M. Ciaramita, A. Gangemi, E. Ratsch, J. Saric and I. Rojas, *Unsupervised Learning of Semantic Relations for Molecular Biology Ontologies*, in: P. Buitelaar, P. Cimiano, eds., Bridging the Gap between Text and Knowledge: Selected Contributions to Ontology Learning and Population from Text, Frontiers in Artificial Intelligence and Applications Series, IOS Press, *this volume*.

[7] A. Schutz, P. Buitelaar, *RelExt: A Tool for Relation Extraction from Text in Ontology Extension*, in: Proceedings of the 4th International Semantic Web Conference, (ISWC) pp. 593–606, 2005.

[8] R. Navigli and P. Velardi, *From Glossaries to Ontologies: Learning the Structure Hidden in the Text*, in: P. Buitelaar, P. Cimiano, eds., Bridging the Gap between Text and Knowledge: Selected Contributions to Ontology Learning and Population from Text, Frontiers in Artificial Intelligence and Applications Series, IOS Press, *this volume*.

[9] M. Poesio and A. Almuhareb, *Extracting concept descriptions from the Web: the importance of attributes and values*, in: P. Buitelaar, P. Cimiano, eds., Bridging the Gap between Text and Knowledge: Selected Contributions to Ontology Learning and Population from Text, Frontiers in Artificial Intelligence and Applications Series, IOS Press, *this volume*.

[10] A. Mädche and S. Staab, *Discovering Conceptual Relations from Text*, in: Proceedings of the European Conference on Artificial Intelligence (ECAI), pp. 321–325, 2000.
[11] M. Berland and E. Charniak, *Finding parts in very large corpora*, in: Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL), 1999.
[12] M. Hearst, *Automatic acquisition of hyponyms from large text corpora*, in: Proceedings of the 14th International Conference on Computational Linguistics, pp. 539-545, 1992.
[13] D. Faure and C. Nedellec, *Knowledge Acquisition of Predicate Argument Structures from Technical Texts Using Machine Learning: The System ASIUM*, in: Proceedings of the European Knowledge Acquisition Workshop (EKAW), pp. 329–334, 1999.
[14] S. A. Caraballo, *Automatic construction of a hypernym-labeled noun hierarchy from text*, in: Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 120–126, 1999.
[15] P. Cimiano and S. Staab, *Learning Concept Hierarchies from Text with a Guided Agglomerative Clustering Algorithm*, in: Proceedings of the ICML 2005 Workshop on Learning and Extending Lexical Ontologies with Machine Learning Methods, 2005.
[16] P. Cimiano, A. Hotho and S. Staab, *Learning Concept Hierarchies from Text Corpora using Formal Concept Analysis*, in: Journal of Artificial Intelligence Research (JAIR), 24, pp. 305–339, 2005.
[17] P. Cimiano and J. Völker, *Towards large-scale, open-domain and ontology-based named entity classification*, in: G. Angelova, K. Bontcheva, R. Mitkov, and N. Nicolov, editors, Proc. of the International Conference on Recent Advances in Natural Language Processing (RANLP), pp. 166–172, Borovets, Bulgaria, September 2005, INCOMA Ltd.
[18] P. Cimiano, S. Handschuh and S. Staab, *Towards the self-annotating web*, in: Proceedings of the 13th World Wide Web Conference, pp. 462–471, 2004.
[19] O. Etzioni, M. J. Cafarella, D. Downey, S. Kok, A.-M. Popesu, T. Shaked, S. Soderland, D.S. Weld and A. Yates, *Web-scale information extraction in KnowItAll (preliminary results)*, in: Proceedings of the 13th World Wide Web Conference, pp. 100–110, 2004.
[20] J. Völker, P. Hitzler and P. Cimiano, *Acquisition of OWL DL Axioms from Lexical Resources*, in: Proceedings of the 4th European Semantic Web Conference (ESWC'07), pp. 670–685, Springer, June 2007.
[21] D. Lin, *Dependency-based evaluation of minipar*, in: Proceedings of the Workshop on the Evaluation of Parsing Systems, 1998.
[22] P. Haase and J. Völker, *Ontology Learning and Reasoning – Dealing with Uncertainty and Inconsistency*, in: P.C.G. da Costa, K.B. Laskey, K.J. Laskey, M. Pool, Proceedings of the Workshop on Uncertainty Reasoning for the Semantic Web (URSW'05), pp. 45–55, November 2005.
[23] P. Cimiano and J. Völker, *Text2Onto – A Framework for Ontology Learning and Data-driven Change Discovery*, in: Andres Montoyo, Rafael Munoz, Elisabeth Metais, Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB'05), volume 3513 of Lecture Notes in Computer Science, pp. 227–238, Springer, Alicante, Spain, June 2005.
[24] Z. Wu and M. Palmer, *Verb semantics and lexical selection*, in: 32nd Annual Meeting of the Association for Computational Linguistics, pp. 133–138, New Mexico, 1994.
[25] C. Fellbaum, *WordNet, an electronic lexical database*, MIT Press, 1998.
[26] Z. Harris, *Distributional structure*, in: J. Katz, editor, The Philosophy of Linguistics, pp. 26–47, New York, 1985, Oxford University Press.
[27] J. Völker, D. Vrandecic, Y. Sure and A. Hotho, *Learning Disjointness*, in: Proceedings of the 4th European Semantic Web Conference (ESWC'07), pp. 175–189, Springer, June 2007.
[28] M. Poesio, T. Ishikawa, S. Schulte im Walde and R. Vieira, *Acquiring lexical knowledge for anaphora resolution*, in: Proceedings of the 3rd Conference on Language Resources and Evaluation, 2002.
[29] D.A. Cruse, *Lexical Semantics*, Cambridge Textbooks in Linguistics, Cambridge University Press, New York, 1986.
[30] N. Guarino and C. A. Welty, *An Overview of OntoClean*, in: S. Staab and R. Studer, eds., The Handbook on Ontologies, pp. 151–172, Springer, 2004.
[31] A. Motro and P. Smets, *Uncertainty Management In Information Systems*, Springer, 1997.
[32] P. Haase and G. Qi, *An analysis of approaches to resolving inconsistencies in DL-based ontologies*, in: Proceedings of the International Workshop on Ontology Dynamics (IWOD'07), June, 2007.
[33] S. Schlobach, *Debugging and semantic clarification by pinpointing*, in: Proceedings of the 2nd European Semantic Web Conference (ESWC'05), volume 3532 of LNCS, pp. 226–240, Springer, 2005.
[34] C. Brewster, H. Alani, S. Dasmahapatra and Y. Wilks, *Data Driven Ontology Evaluation*, in: Proceedings of the Lexical Resources and Evaluation Conference (LREC'04), pp. 641–644, Lisbon, Portugal, 2004.

[35] M. Ehrig, P. Haase, N. Stojanovic, and M. Hefke, *Similarity for ontologies - a comprehensive framework*, in: Proceedings of the 13th European Conference on Information Systems, May 2005.
[36] J. Völker, D. Vrandecic and Y. Sure, *Automatic Evaluation of Ontologies (AEON)*, in: Y. Gil, E. Motta, V. R. Benjamins, M. A. Musen, Proceedings of the 4th International Semantic Web Conference (ISWC'05), volume 3729 of LNCS, pp. 716–731, Springer Verlag Berlin-Heidelberg, November 2005.
[37] S. Rudolph, *Exploring relational structures via FLE*, in: Wolff, K.E., Pfeiffer, H.D., Delugach, H.S., eds.: Conceptual Structures at Work, 12th International Conference on Conceptual Structures, volume 3127 of LNCS, Springer, pp. 196–212, 2004.
[38] B. Ganter and R. Wille, *Formal Concept Analysis – Mathematical Foundations*, Springer, Berlin, 1999.
[39] D. Lin and P. Pantel: *DIRT – discovery of inference rules from text*, in: Knowledge Discovery and Data Mining, pp. 323–328, 2001.