D3.3.3 Data-driven Change Discovery

Johanna Völker, Denny Vrandečić and York Sure
(University of Karlsruhe)

Abstract.
EU-IST Integrated Project (IP) IST-2003-506826 SEKT
Deliverable D3.3.3 (WP3.3)
In this deliverable we present our work on ontology learning and evaluation performed in task 'T3.3 Data-driven Change Discovery'.
Keyword list: Data-driven Change Discovery, Ontology Learning, Ontology Management

Document Id.	SEKT/2006/D3.3.3/v1.2
Project	SEKT EU-IST-2003-506826
Date	December 15, 2006
Distribution	public

SEKT Consortium

This document is part of a research project partially funded by the IST Programme of the Commission of the European Communities as project number IST-2003-506826.

British Telecommunications plc.
Orion 5/12, Adastral Park
Ipswich IP5 3RE, UK
Tel: +44 1473 609583, Fax: +44 1473 609832
Contact person: John Davies
E-mail: john.nj.davies@bt.com

Jozef Stefan Institute
Jamova 39
1000 Ljubljana, Slovenia
Tel: +386 1 4773 778, Fax: +386 1 4251 038
Contact person: Marko Grobelnik
E-mail: marko.grobelnik@ijs.si

University of Sheffield
Department of Computer Science
Regent Court, 211 Portobello St.
Sheffield S1 4DP, UK
Tel: +44 114 222 1891, Fax: +44 114 222 1810
Contact person: Hamish Cunningham
E-mail: hamish@dcs.shef.ac.uk

Intelligent Software Components S.A.
Pedro de Valdivia, 10
28006 Madrid, Spain
Tel: +34 913 349 797, Fax: +49 34 913 349 799
Contact person: Richard Benjamins
E-mail: rbenjamins@isoco.com

Ontoprise GmbH
Amalienbadstr. 36
76227 Karlsruhe, Germany
Tel: +49 721 50980912, Fax: +49 721 50980911
Contact person: Hans-Peter Schnurr
E-mail: schnurr@ontoprise.de

Vrije Universiteit Amsterdam (VUA)
Department of Computer Sciences
De Boelelaan 1081a
1081 HV Amsterdam, The Netherlands
Tel: +31 20 444 7731, Fax: +31 84 221 4294
Contact person: Frank van Harmelen
E-mail: frank.van.harmelen@cs.vu.nl

Siemens Business Services GmbH & Co. OHG
Otto-Hahn-Ring 6
81739 Munich, Germany
Tel: +49 89 636 40 225, Fax: +49 89 636 40 233
Contact person: Dirk Ramhorst
E-mail: dirk.ramhorst@siemens.com

Empolis GmbH
Europaallee 10
67657 Kaiserslautern, Germany
Tel: +49 631 303 5540, Fax: +49 631 303 5507
Contact person: Ralph Traphöner
E-mail: ralph.traphoener@empolis.com

University of Karlsruhe, Institute AIFB
Englerstr. 28
D-76128 Karlsruhe, Germany
Tel: +49 721 608 6592, Fax: +49 721 608 6580
Contact person: York Sure
E-mail: sure@aifb.uni-karlsruhe.de

University of Innsbruck
Institute of Computer Science
Technikerstraße 13
6020 Innsbruck, Austria
Tel: +43 512 507 6475, Fax: +43 512 507 9872
Contact person: Jos de Bruijn
E-mail: jos.de-bruijn@deri.ie

Kea-pro GmbH
Tal
6464 Springen, Switzerland
Tel: +41 41 879 00, Fax: 41 41 879 00 13
Contact person: Tom Bösser
E-mail: tb@keapro.net

Sirma Group Corp., Ontotext Lab
135 Tsarigradsko Shose
Sofia 1784, Bulgaria
Tel: +359 2 9768 303, Fax: +359 2 9768 311
Contact person: Atanas Kiryakov
E-mail: naso@sirma.bg

Universitat Autonoma de Barcelona
Edifici B, Campus de la UAB
08193 Bellaterra (Cerdanyola del Vallès)
Barcelona, Spain
Tel: +34 93 581 22 35, Fax: +34 93 581 29 88
Contact person: Pompeu Casanovas Romeu
E-mail: pompeu.casanovas@uab.es

Executive Summary

This deliverable, D3.3.3 Data-driven Change Discovery, deals with two approaches to enable the automation of the evolution of ontologies. First, we present a new version of Text2Onto, a framework for ontology learning from text. The previously existing English version of Text2Onto has been adapted to support the linguistic analysis of Spanish documents, including language-specific algorithms for the extraction of ontological concepts, instances and relations. The adaptation to Spanish stems from the requirements collected from on of the SEKT case studies, i.e. the legal spanish case study. Second, we present an approach for the learning of disjointness axioms. 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. The approach advances significantly the state-of-the-art and reflects a trend towards usage of more expressive ontology languages such as OWL.

Contents

Chapter 1

Introduction

1.1 The SEKT Big Picture

This report is part of the work performed in workpackage (WP) 3 on "Ontology and Metadata Management". It specifically refers to task 'T3.3 Data-driven Change Discovery'. As shown in Figure 1.1 this work is closely related with other technical workpackages in SEKT. The main goal of this workpackage is to enable and to facilitate the setting up and maintenance of semantic knowledge management applications by supporting the complex tasks of managing ontologies and corresponding metadata.

1.2 Motivation

As described in [VS05a] the ontology learning framework Text2Onto has been designed to provide an infrastructure for data-driven change discovery in the SEKT project. In deliverable 3.3.2 [VS05b] we evaluated our initial prototype in the context of the BT Case Study (WP11), and we presented AEON, a framework for the automatic evaluation of ontologies with respect to the OntoClean methodology. This deliverable extends our previous work.

We present two approaches to enable the automation of the evolution of ontologies. As data we use in the SEKT case studies text documents (English and Spanish) from which we first learn ontologies and then evolve them, both in an automatic manner. Our work in the final year of the SEKT project has been driven by the requirements coming from the case studies as well as current scientific trends.

First, we present a new version of Text2Onto, a framework for ontology learning from text. The previously existing English version of Text2Onto has been adapted to support the linguistic analysis of Spanish documents, including language-specific algorithms for the extraction of ontological concepts, instances and relations. The adaptation to Spanish

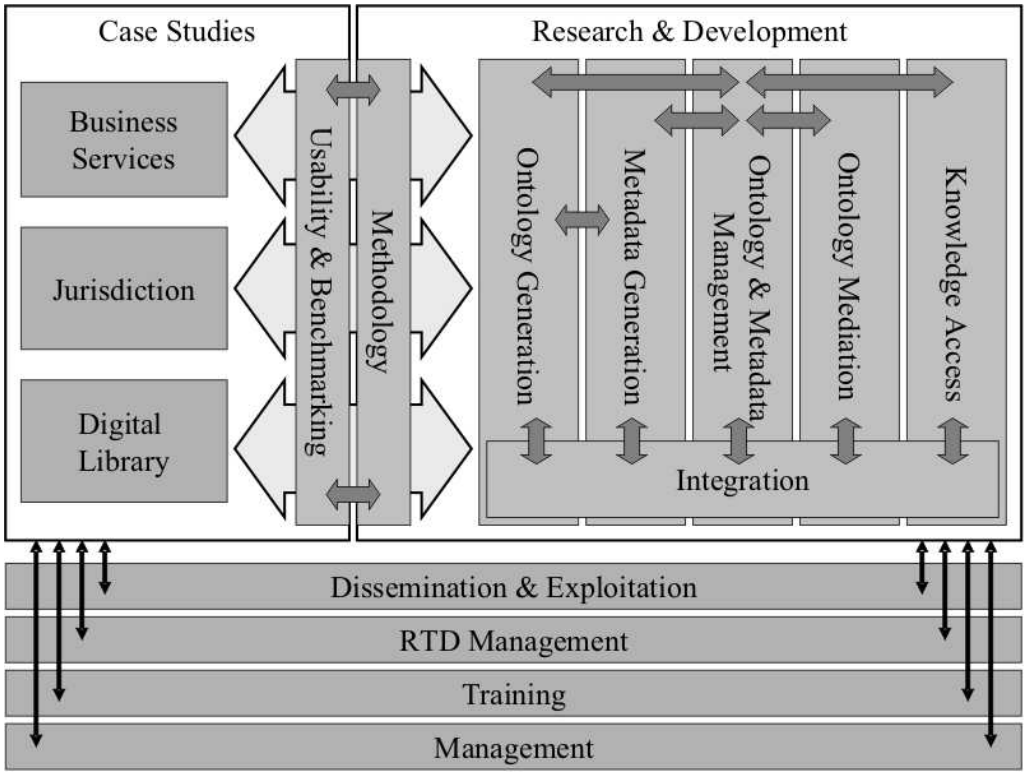

Figure 1.1: The SEKT Big Picture

stems from the requirements collected from on of the SEKT case studies, i.e. the Spanish Legal Case Study (WP10).

Second, we present an approach for the learning of disjointness axioms. Introducing disjointness axioms, for example, greatly facilitates consistency checking, the detection of modelling errors and the automatic evaluation of individuals in a knowledge base with regards to a given ontology. The approach advances significantly the state-of-the-art and reflects a trend towards usage of more expressive ontology languages such as OWL.

1.3 Overview

The following chapter 2 gives an overview of the current status of Text2Onto, particularly the components required for processing Spanish texts (see Section 2.3). In Chapter 3 we describe a classification-based approach for learning disjointness axioms (cf. Section 3.2) along with a detailed evaluation (see Sections 3.3 and 3.4). Finally, chapter 4 concludes with a summary and an outlook to future work.

Chapter 2

Text2Onto

2.1 Introduction

The case study on "Intelligent Integrated Decision Support for Legal Professionals" (WP10) aims at supporting newly appointed judges in Spain by means of iFAQ, an intelligent Frequently Asked Questions. iFAQ relies on several complex ontologies of the legal domain, among them the Ontology of Professional Judicial Knowledge (OPJK) consisting of about 100 classes and more than 500 instances. Building and maintaining this ontology is a tedious and time-consuming task requiring profound knowledge of legal documents and language. Therefore, any kind of automatic support can significantly increase the efficiency of the knowledge acquisition process.

In this chapter we present a new version of Text2Onto[1], a framework for ontology learning from text. The previously existing English version of Text2Onto has been adapted to support the linguistic analysis of Spanish documents, including language-specific algorithms for the extraction of ontological concepts, instances and relations. Since a detailed description of the individual algorithms has already been given in [VS05a] we focus on the key differences between the two versions.

2.2 Linguistic Preprocessing

All of the algorithms being part of the Text2Onto framework largely rely on a combination of machine learning and natural language processing techniques in order to extract ontology entities and relationships from open-domain unstructured text. Since the necessary linguistic analysis is done by means of GATE [CMBT02] it is very flexible with respect to the set of linguistic components used, i.e. the underlying GATE application can be freely configured by replacing existing components or adding new ones such as a deep

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

parser if required. Another benefit of using GATE is the seamless integration of JAPE which provides finite state transduction over annotations based on regular expressions.

- **Tokenizer:** splits text into individual tokens, basically words and punctuation symbols.
- **Sentence splitter:** detects sentence boundaries.
- **Part-of-Speech tagger:** assigns a syntactic category to each token.
- **Lemmatizer:** reduces each token to its lemma, i.e. base form.
- **JAPE transducer** for shallow parsing: identifies chunks of tokens which constitute e.g. noun phrases or verb phrases.

Linguistic preprocessing in Text2Onto starts by tokenization and sentence splitting. The resulting annotation set serves as an input for a POS tagger which in the following assigns appropriate syntactic categories to all tokens. Finally, lemmatizing or stemming (depending on the availability of the regarding processing components for the current language) is done by a morphological analyzer or a stemmer respectively.

In order to improve the quality of the linguistic analysis particularly for Spanish text, we replaced some of the standard GATE components by external resources. The TreeTagger[2] is a POS tagger and lemmatizer developed by the University of Stuttgart which can be adapted to a multitude of languages by means of language-specific parameter files. The following screenshot 2.1 shows the output of the TreeTagger's command line interface. Our GATE wrapper transforms this output into GATE annotations which are available at all subsequent stages in the linguistic processing pipeline.

After the basic linguistic preprocessing is done, an additional JAPE transducer is run over the annotated corpus in order to match a set of particular JAPE patterns for shallow parsing. These JAPE patterns have to take into account the specific structure of Spanish noun phrases, verb phrases and prepositional phrases. For example, Spanish other than English noun phrases may contain adjectives before as well as after the head of the phrase, and a prepositional complement.

In the following sections we describe the core algorithms provided by the Text2Onto framework and their required adaption to the Spanish language.

[2] `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/`

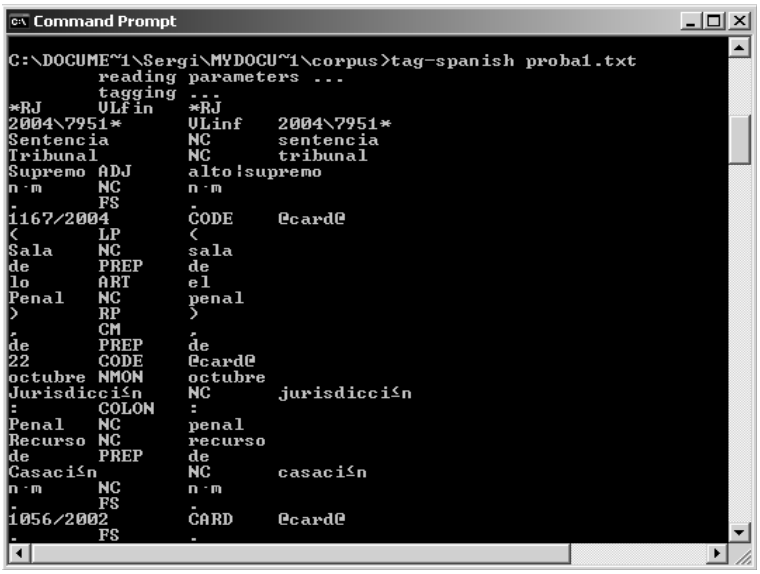

Figure 2.1: TreeTagger

2.3 Algorithms for Spanish

2.3.1 Concepts and Instances

The extraction of concepts and instances relies on a set of JAPE patterns for identifying common and proper noun phrases. Each noun phrase is assigned a relevance value before being mapped to a new or existing class in the ontology. The relevance values with respect to the particular domain are computed by means of statistical measures such as average TFIDF or entropy. While these measures are in principle language independent, the structure of Spanish noun phrases requires specific treatment. We therefore had to develop a number of new JAPE patterns for shallow parsing to be matched during the linguistic preprocessing phase.

2.3.2 SubclassOf Relations

In the previous version of Text2Onto mainly three algorithms were used for extracting subclassOf relationships from English text. As described in the following sections all of them had to be adapted to the requirements of the Spanish language. Figure 2.2 shows the results of the Spanish concept classification which is described in the following sections.

Please note that the screenshot as well as most of the examples in this chapter were created from a corpus of web documents about ontologies and the semantic web, since the terminology of this domain is more easily accessible to non Spanish speaking people. Evaluation results and detailed examples relating to original legal case study data are

given in [PC06].

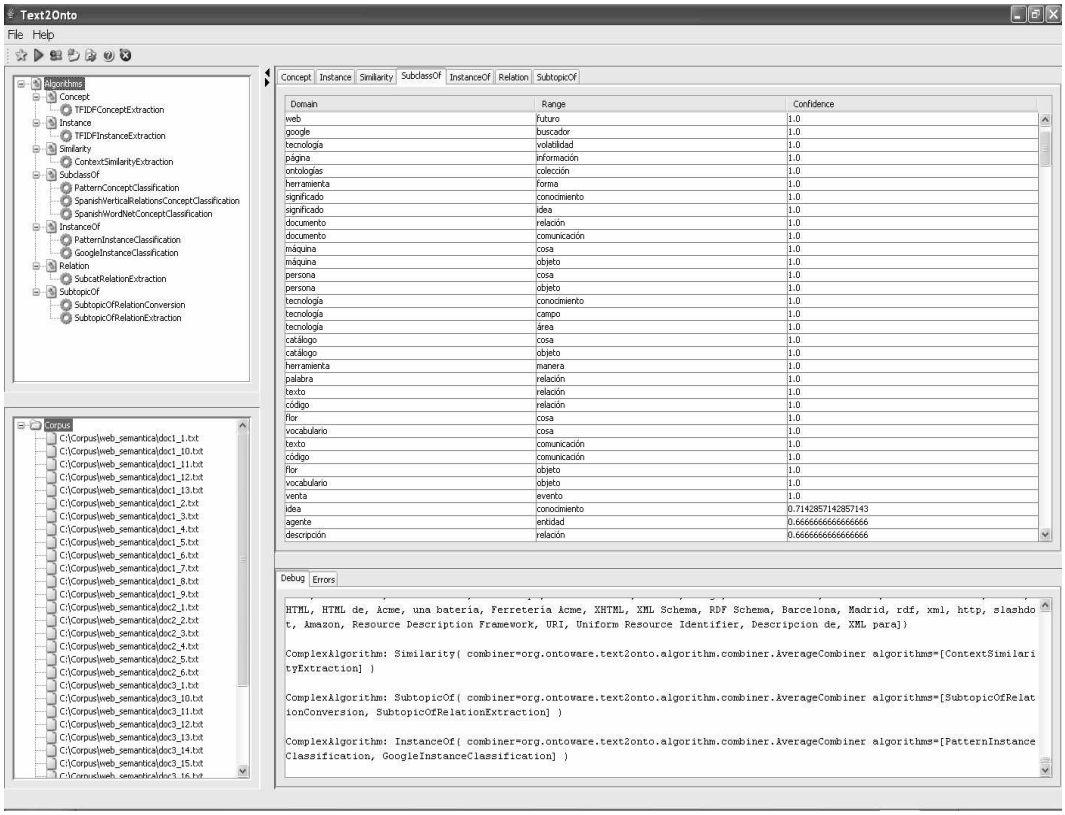

Figure 2.2: Results (subclassOf)

Patterns

The pattern-based concept classification algorithm relies upon a number of lexico-syntactic patterns indicating hyponymy relationships. SubclassOf relations are generated based on this evidence and annotated with a confidence value that corresponds to the normalized frequency of pattern occurrences.

$NP_{superclass}$ *como* (*por ejemplo*)? $NP_{subclass}$

$NP_{subclass}$ (*son*|*es*|*eran*|*era*) $NP_{superclass}$

$NP_{subclass}$ (*y*|*o*) (*otros*|*otras*|*demás*) $NP_{superclass}$

$NP_{superclass}$ (*incluiendo*|*especialmente*) $NP_{subclass}$

$NP_{superclass}$ *tal*|*tales* *como* $NP_{subclass}$

WordNet

For any given pair of classes the WordNet-based concept classification aims to find evidence for a hyponymy relationship between the corresponding terms by querying WordNet.

Since the standard version of WordNet provided by Princeton University has been developed particulary for the English language, it is unsuitable for processing Spanish texts. We therefore integrated a Spanish version of WordNet which is developed and maintained by the natural language processing group of the Technical University of Catalonia (UPC).

Vertical Relations Heuristic

The vertical relations heuristic generates subclassOf relations from composite noun phrases, assuming that the class denoted by the whole phrase is subsumed by the class which is represented by its head. For example, from the noun phrase "buscador semántico"[3] ("semantic search engine") the algorithm would conclude that the class *BuscadorSemantico* is subsumed by *Buscador*.

2.3.3 InstanceOf Relations

Basically three algorithms are available for learning instanceOf relationships from Spanish texts.

Patterns

The pattern-based extraction of instanceOf relationships is very similar to the concept-based concept classification described in Section 2.3.2. Some of the lexico-syntactic patterns used for detecting concept instantiations are listed below.

$PNP_{instance}$, NP_{class} ,

$PNP_{instance}$ (NP_{class})

NP_{class} *como* (*por ejemplo*)? $PNP_{instance}$

$PNP_{instance}$ (*son*|*es*|*eran*|*era*) NP_{class}

$PNP_{instance}$ (*y*|*o*) (*otros*|*otras*|*demás*) NP_{class}

NP_{class} (*incluyendo*|*especialmente*) $PNP_{instance}$

NP_{class} *tal*|*tales* *como* $PNP_{instance}$

[3]Please note that the alternative spelling "buscador semantico" will not be recognized as denoting the same class by the current version of Text2Onto. This bug will be fixed for the next version.

Google

In line with [CLS05] we implemented an approach to obtaining evidence for instanceOf relations by online pattern matching. For each instance lexically represented by $PNP_{instance}$ the algorithm poses a number of Google queries similar to those described in the previous section.

"*como* $PNP_{instance}$"
"$PNP_{instance}$ *es un*"
"$PNP_{instance}$ *es una*"

The results returned for each of these queries are then analysed in order to determine possible fillers for the open position in the regarding pattern. For the first query template, for example, the filler must be a noun phrase directly preceding the phrase matched by the query.

Context-based Similarity

The assumption underlying the context-based instance classification is that each instance belongs to the class which is semantically most similar.

In order to compute the semantic similarity Text2Onto exploits the distributional hypothesis by Harris which basically states that the senses of two words, i.e. concepts, are similar to the degree the words share lexical context. Lexical context in its most simple form is a vector consisting of all the words which co-occur (e.g. in the same sentence or token window) with the words representing the class or instance of interest. For the new version of Text2Onto we implemented a more sophisticated context extraction based on both lexical and syntactic features.

The context vector of each instance is compared to all context vectors of concepts in the ontology by means of the cosine measure. If the similarity of the context vectors is above a certain threshold the instance classification algorithms assumes the instance to instantiate that particular concept. Further details regarding different types of context features and similarity measures for instance classification are given by [CV05c].

2.3.4 Non-taxonomic Relations

For the extraction of non-taxonomic relationships Text2Onto relies upon subcategorization frames, i.e. predicate argument structures consisting of verbs and prepositional or nominal complements, enriched by ontological knowledge and statistical information. Confidence values for each relationship are computed based on the

number of instantiations of that particular frame found in the corpus. Example: *incluir*(*Ontología*, *Definición*).

2.4 Lessons Learned and Conclusion

Adapting Text2Onto to the requirements of the Spanish language confronted us with a number of unexpected technical challenges. Some of them had to do with syntactic and semantic particularities of the Spanish language, e.g. with respect to prepositional complements in noun phrases. Others were related to character encoding, compatibility of different versions of WordNet and the adaptation of new linguistic processing components.

On the other hand, we found that a flexible and extensible language processing framework such as GATE is of great use if multilinguality is required by the application. This flexibility made it possible to integrate specialized linguistic components for the Spanish language into Text2Onto without much effort, and significantly sped up the implementation process.

The final evaluation of Text2Onto in the Legal Case Study is still ongoing. However, initial experiments indicate that Text2Onto significantly helps in building ontologies from a given collection of Spanish documents (see upcoming SEKT deliverable 10.4.1 [PC06]). And we are confident that it can be a valuable part in an iterative process of learning and manual refinement.

Chapter 3

Learning Disjointness

3.1 Introduction

An increasing number of applications benefits from light-weight ontologies, or, to put it differently, *"a little semantics goes a long way"* (Jim Hendler). Our experience in building ontology-based systems indicates, however, that adding more expressivity in a controlled manner can reap further benefits. 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.

In description logics two classes are considered as disjoint *iff* their *taxonomic overlap*, i.e. the set of common individuals, *must* be empty. This does not include classes with actual extensions that coincidentally do not have common individuals, for instance *Woman* and *US President*, but only those where the common subset must be empty in all possible worlds – like, for example, *Woman* and *Car*.

Disjointness allows for far more expressive and meaningful ontologies, as shown exemplary in the following. An ontology language with the expressivity of RDFS does not constrain the possible assertions in any way. Even after we set up an ontology defining terms like *Book, Student* and *University*, stating that *John* is both a *Student* and a *University* is logically perfectly viable and would not be recognized as an error by the ontology management system. Only if we define these classes as being disjoint, a reasoner will be able to infer the error in the above ontology, guaranteeing that particular constraints are met by the knowledge base and a certain quality of facts is achieved – thus raising the quality of the whole ontology-based system [Sch05].

Despite the obvious importance of stating disjointness among classes, many of today's ontologies do not contain any disjointness axioms. In fact, a survey of 1,275 ontologies [Wan06] recently found only 97 of them to include disjointness axioms. We can only speculate about the reasons, but it is very likely that ontology engineers often forget to introduce disjointness axioms, simply because they are not aware of the fact that classes

which are not explicitly declared to be disjoint will be considered as overlapping. Particularly, inexperienced users usually assume the semantics of partitions, or even complete partitions, when they build a subsumption hierarchy (see [RDH+04]). Also, as the size of an ontology is a major cost driver for ontologies [BTS06], the manual engineering and addition of the axioms actually costs more time, and thus money.

Therefore, we believe that an approach to automatically introduce disjointness axioms into an ontology would be a valuable addition to any ontology learning or engineering framework. The principle feasibility of learning disjointness based on simple lexical evidence has already been shown by [HV05]. 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.

We performed an extensive survey in order to collect experience with modelling disjoint classes, and identified several problems frequently encountered by users who try to introduce disjointness axioms. Based on the results of our 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. 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 more than 10,000 disjointness axioms provided by 30 human annotators. Due to the encouraging evaluation results we are confident that our implementation can be used, for example, to extend state-of-the-art ontology learning systems, to support ontology debugging [Sch05], or to evaluate manually added disjointness axioms.

The survey also showed that deciding if two classes are disjoint is far from trivial. Although experts have a higher agreement on disjointness than non-expert users, their agreement is still lower than we expected. Discussing these problematic formalizations, we uncovered a number of problems humans have with formal disjointness.

In this paper, we will, in Section 3.2, first present the features we have used in order to automatically learn disjointness axioms. Section 3.3 describes the set up and execution of the experiments we conducted in order to train a classifier and evaluate the results of our implementation (Section 3.4). We close with an overview of related work in Section 3.5 and a summary of the key contributions and remaining open questions in Section 3.6.

3.2 Features for Learning Disjointness

Assuming that there is not the one and only approach to determine the disjointness of two classes in an ontology, we developed a variety of different methods to obtain evidence for or against disjointness from different sources. The features delivered by these methods will help us to train a classifier which is able to distinguish between disjoint and non-disjoint classes.

In the next subsections – after some preliminary remarks regarding the notation throughout this paper and some basic assumptions we made for the feature extraction process – we motivate and describe the construction of ten highly selective features. A machine learning classifier will then be trained on a set of manual annotations as described in Section 3.4.

In this paper we adopt the OWL ontology model, although we do not restrict our approach to OWL. Any ontology model that allows to state disjointness between two classes can be used with all the methods described in this paper.

Preliminaries: The methods the extraction of classification features are provided with an unsorted list of all the pairs previously tagged by human annotators. In the following the set of pairs will be denoted by $P = \{p_1, ...p_n\}$ for $0 \leq n \leq |C|^2$, where C is the set of all classes in the ontology. Each pair $p_k = (c_{k_1}, c_{k_2})$ consists of two classes $c_{k_1}, c_{k_2} \in C$ and $c_{k_1} \neq c_{k_2}$. The confidence of the system in c_{k_1} and c_{k_2} being (not) disjoint is denoted by *conf*$(p_k, +)$ or *conf*$(p_k, -)$ respectively.

All methods are allowed to look up these classes within their semantic context, i.e. the domain **ontology** they have been extracted from (see Section 3.3.1). And finally, as additional sources of background knowledge, the methods may make use of a **corpus** of textual resources associated with the ontology. We automatically selected a subset of 957 documents from the Reuters corpus[1] [RSW02]. For efficiency reasons we only chose those documents with at least 20 occurrences of classes from the ontology.

It is important to mention, that we assume "meaningful" labels for all classes in the ontology, i.e. labels which may be understood by humans even without knowing the whole taxonomy. This assumption is particularly relevant for all methods which make use of textual resources such as the pattern-based disjointness extraction (cf. Section 3.2.4), the computation of extensional overlap with respect to Del.icio.us[2] and the algorithms for learning taxonomic relationships (see Section 3.2.1).

3.2.1 Taxonomic Overlap

In description logics two classes are disjoint *iff* their *taxonomic overlap*, i.e. the set of common individuals, is empty. Because of the open world assumption in OWL, these individuals do not necessarily have to *exist* in the ontology. 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.

Therefore, 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 documents** tagged with the corresponding class labels.

[1] `http://trec.nist.gov/data/reuters/reuters.html`
[2] `http://del.icio.us/`

Ontology

Both, individuals and subclasses can be imported from an ontology (see Section 3.3.1) 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 ontology learning framework. A detailed description of these algorithms can be found in [CV05a]. All taxonomic relationships – learned and imported ones – are associated with rating annotations $r_{subclass-of}$ (or $r_{instance-of}$ respectively) indicating the certainty $x > 0$ of the underlying ontology learning framework in the correctness of its results[3].

$$r_{subclass-of}(c_1, c_2) = \begin{cases} x & c_1 \texttt{ subclass-of } c_2 \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

The following formula defines the confidence $conf(p, -)$ for a pair $p = (c_1, c_2)$ to be **not** disjoint based on the taxonomic overlap of c_1 and c_2 with respect to common **subclasses** (the same for **instance**):

$$conf(p, -) = \frac{\sum_{c \in sub_1 \cap sub_2} (r_{subclass-of}(c, c_1) \cdot r_{subclass-of}(c, c_2))}{\sum_{c \in sub_1} r_{subclass-of}(c, c1) + \sum_{c \in sub_2} r_{subclass-of}(c, c_2)} \tag{3.2}$$

where sub_i denotes the set of subclasses of c_i.

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 in total $|Y| = 17,362,212$ triples. The idea underlying the use del.icio.us in this case is similar to the one described in subsection 3.2.4: If two labels are frequently used to tag the same resource they are likely to be disjoint, because users tend to avoid redundant labeling of documents.

$$conf(p, -) = \frac{|\{d | c_1 \in t(d), c_2 \in t(d)\}|}{\sum_{c \in C} |\{d | c_1 \in t(d), c \in t(d)\}| + \sum_{c \in C} |\{d | c_2 \in t(d), c \in t(d)\}|} \tag{3.3}$$

where $t(d)$ is 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.

[3] For imported relationships the confidence is 1.0.

3.2.2 Subsumption

If one class is a subclass of the other we assume the two classes of a pair $p = (c_1, c_2)$ to be **not** disjoint with a confidence equal to the likeliness associated with the `subclass-of` relationship (c.f. Section 3.2.1).

$$conf(p, -) = \max(r_{subclass-of}(c_1, c_2)), r_{subclass-of}(c_2, c_1))) \tag{3.4}$$

3.2.3 Semantic Similarity

The assumption of a direct correspondence between the semantic similarity of two classes are their likeliness to be disjoint led to the development of three methods: The first one implements the similarity measure described by [WP94] to compute the semantic similarity sim of two classes c_1 and c_2 with respect to **WordNet** [Mil95].

$$conf(p, -) = sim(s_1, s_2) = \frac{2 * depth(lcs(s_1, s_2))}{depth(s_1) + depth(s_2)} \tag{3.5}$$

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

The second method measures the distance of c_1 and c_2 with respect to the given background **ontology** (see Section 3.3.1) by computing the minimum length of a path p of `subclass-of` relationships connecting c_1 and c_2.

$$conf(p, +) = \min_{p \in paths(c_1, c_2)} length(p) \tag{3.6}$$

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 [CV05b] we exploit Harris' distributional hypothesis [Har68] 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 prelimaries 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 using a modified version of the tf-idf formula:

Let $v_i = (f_1^i ... f_n^i)$ be the context vector of class c_i where each f_j^i, $n \geq 1$ is the frequency of token j in the context of c_i. Then we define $TF(f_j^i) = \sum_{d \in doc(c_i)} freq(f_j^i, d)$ and $N = |doc(c_i)|$ and $DF = |doc(c_i) \cap doc(f_j^i)|$, where $doc(t)$ is the set of documents

containing term t and $freq(t,d)$ is the frequency of term t in document d. And finally, we get $TFIDF(f_j^i) = TF(f_j^i) \cdot \log(\frac{N}{DF})$.

Given the *weighted* context vectors v_1' and v_2' the confidence in c_1 and c_2 being **not** disjoint is defined as $conf(p,-) = \cos(v_1', v_2')$.

3.2.4 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 [HV05]. The underlying assumption in there is that 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.

we would conclude that *pig*, *cow*, *horse*, *duck* and *dog* are 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 for the disjointness of two concepts 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:

$$conf(p,+) = \frac{freq(c_1,c_2)}{\sum_{j\neq 1} freq(c_1,c_j) + \sum_{i\neq 2} freq(c_i,c_2)} \tag{3.7}$$

where $freq(c_i,c_j)$ is the number of patterns providing evidence for the disjointness of c_i and c_j with $0 \leq i,j \leq |C|^2$ and $i \neq j$.

3.2.5 OntoClean

In [VVS05] we introduced AEON, an approach to automatically evaluate ontologies according to the OntoClean methodology [GW00]. The basic idea is to use a pattern-based approach on top of the Web (and other textual data sources) for annotating classes of a

given ontology with the OntoClean properties such as unity, identity and rigidity. Parts of the approach can be reused for learning disjointness axioms.

Two classes are disjoint if they have incompatible unity or identity criteria. This implies that a class carrying anti-unity ($\sim$U) must be disjoint of a class carrying unity (+U) – and similarly for identity. Since we use the same subset of the PROTON ontology as in our AEON experiments (c.f. Section 3.3.1), we can rely on the manual OntoClean taggings we collected earlier for the evaluation of AEON.

$$conf(p,+) = \begin{cases} 1 & \text{if } c_1 \text{ tagged with } \phi\Omega,\ c_2 \text{ tagged with } \psi\Omega, \\ & \text{for } \Omega \in \{U, I\},\ \phi, \psi \in \{\sim, +\},\ \phi \neq \psi \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

3.2.6 Meta Algorithm

The meta algorithm considers superclasses known to be disjoint (from previously computed confidence values) and propagates this information downwards in the taxonomic hierarchy. For $p = (c_1, c_2)$ the confidence for c_1 and c_2 being disjoint is computed as follows:

$$conf(p,+) = \frac{\sum_{p^s}(conf(p^s,+) - conf(p^s,-))}{|super(c_1)| \cdot |super(c_2)|} \tag{3.9}$$

where $p^s = (c_1^s, c_2^s)$ with $c_i^s \in \{c | subclass - of(c_i, c)\}$ for $i \in \{1, 2\}$ and $subclass - of(c_i, c_j)$ being the `subclass-of` relationship between c_i and c_j. Moreover, $super(c)$ denotes the set of superclasses of c.

3.3 Experiment: Human Annotation of Disjointness

We thoroughly evaluated our approach by performing a comparison of learned disjointness axioms against a large number of manually created ones to calculate (among other things) the degree of overlap. This section describes the generation of the evaluation dataset consisting of 2000 pairs of classes tagged by 30 annotators, and discusses methodological aspects related to the manual creation of disjointness axioms.

The structure of this section is as follows: First, we give a brief overview of the PROTON ontology which serves as a basis for our experiments (Section 3.3.1). In Section 3.3.2 we describe the creation of a gold standard which is later used to evaluate our classification-based approach (Section 3.4). Finally, sections 3.3.3 and 3.3.4 analyse the main difficulties encountered by our human annotators, in order to illustrate the difficulty of the task.

Please note that the complete dataset is available for download from `http://www.aifb.uni-karlsruhe.de/WBS/jvo/data/disjointness-111206.zip`.

3.3.1 Ontology

As a basis for the creation of the evaluation datasets and as background knowledge for the ontology learning algorithms we took a subset (*system*, *top* and *upper* module) of the freely available PROTON ontology (PROTo ONtology)[4]. In total our subset of PROTON contains 266 classes, 77 object properties, 34 datatype properties and 1388 siblings.

PROTON is a basic upper-level ontology to facilitate the use of background or pre-existing knowledge for automatic metadata generation. PROTON covers the general concepts necessary for a wide range of tasks, including semantic annotation, indexing, and retrieval of documents. The design principles can be summarized as follows (as described in [TKM04]) (i) domain-independence; (ii) light-weight logical definitions; (iii) alignment with popular standards; (iv) good coverage of named entities and concrete domains (i.e. people, organizations, locations, numbers, dates, addresses).

3.3.2 Evaluation Setting: Manual Taggings

To be able to compare the results of our trained model with the results generated by manual annotation we created a dataset consisting of 2000 pairs of classes as follows: First, we manually selected 200 (potentially) *non-disjoint* pairs from the ontology (see Section 3.3.1), since we assumed the set of non-disjoint pairs to constitute a weak minority class (which would have hampered the construction of a good model for our classifier). Then, we randomly chose 500 *siblings* – which constitute a subset of the data, which is of particular interest from a practical and theoretical aspect. And finally, we added another 1300 pairs chosen *randomly* without any selection criteria.

Once the dataset was complete, each pair was randomly assigned to 6 different people – 3 from each of two groups, the first one consisting of PhD students from our institute (all of them professional "ontologists"), the second being composed of under-graduate students without profound knowledge in ontological engineering. Each of the annotators was given between 385 and 406 pairs along with natural language descriptions of the classes whenever those were available. Possible taggings for each pair were $+$ (disjoint), $-$ (not disjoint) and $?$ (unknown). The result were two datasets A and B for "ontologists" and "students" and a third dataset C which was created by merging A and B (cf. table 3.1a)). Dataset D is a subset of C consisting of all siblings, whereas E contains all those pairs of classes which were randomly selected.

In order to get cleaner and less ambiguous training data for our classification model (see Section 3.4) we computed the *majority votes* for all the above mentioned datasets

[4]PROTON is available at `http://proton.semanticweb.org/`.

by considering the individual taggings for each pair (3 in the case of A and B, and 6 for C). If at least 50% (or 100% respectively) of the human annotators agreed upon $+$ or $-$ this decision was assumed to be the majority vote for that particular pair. In case of equally many positive and negative taggings, the majority vote was defined as $?$ or *unknown*. These pairs were not used for training purposes. Some statistical properties of the majority vote datasets are given by table 3.3.

3.3.3 Analysis of Human Annotations

In order to determine how difficult it is for humans to tag pairs of classes as being disjoint or not we measured the human agreement within and across different parts of the dataset (c.f. table 3.1). Table 3.2 shows the average agreement among the individual taggers, i.e. the average maximum ratio of annotators who agreed upon the same tag for a pair of classes. By analysing the figures we find that the average agreement for D is significantly lower than the agreement for any of the other datasets – which seems to imply that pairs of siblings (classes with a common direct superclass) are much more difficult to tag for human annotators than randomly chosen pairs of classes. This might be due to the fact that it is comparably hard to determine the differences between the intension and extension of classes which are semantically very close.

Table 3.1: Evaluation Datasets

ID	Dataset	Annotators	Tags per Pair	Pairs
A	Experts	15	3	2000
B	Students	15	3	2000
C	All	30	6	2000
D	Siblings	30	6	541
E	Random	30	6	1300

Table 3.2: Tagged Pairs (Individual)

Dataset	Individual Taggings					
	$+$	$-$	$?$	all	$-/+$	avg. agree.
A	3849	2007	144	6000	0.521	0.869
B	3881	2106	13	6000	0.543	0.858
$avg.$	3865.0	2056.5	78.5	6000	0.532	0.864
C	7730	4113	157	12000	0.532	0.824
D	1362	1822	62	3246	1.338	0.754
E	6166	1554	80	7800	0.252	0.853

In addition to the computation of the agreement within each of the datasets, we also tried to capture communalities and differences between the taggings of people from the two groups of annotators – ontologists (A) and students (B).

First, we measured the average agreement of the individual tagging of the experts with the majority vote 100% of the students and vice versa. The figures – 0.852 for the agreement between A and the majority vote of B, and a slightly lower value of 0.834 for the agreement between B and the majority vote of A – indicate that, maybe due to the relatively higher disagreement among the students (see table 3.2b)), those tend to agree mainly on very evident cases of disjointness.

The hypothesis that there is a considerable number of pairs which are comparably easy to tag, thus provoking a high agreement, is supported by the figures we get for the agreement among the majority votes 100% (0.964) and 50% (0.793) of A and B.

And finally, we completed our analysis of the annotation results by inspecting concrete examples of differently tagged pairs. Table 3.4 and 3.5 contain all pairs of classes which were assigned different tags by the majority votes 100% (which means that all 3 annotators of A or B agreed upon each tag) of experts and students. An extensive discussion of the differences which tries to explain some of the problems the human annotators encountered can be found in the following section.

3.3.4 Discussion

During the creation of the human annotations, we had the chance to study the problems humans face when using disjointness. Even in the taggings of the experts group – consisting of post-graduates all involved in Semantic Web research – the overlap of the taggings was lower than expected (cf. Section 3.3.3). Table 3.4 and 3.5 show all pairs where all experts agreed on one tagging, and all students agreed on the other. Based on an analysis of the taggings and subsequent discussions with the taggers, we identified several types of problems regarding disjointness:

Table 3.3: Tagged Pairs (Majority Vote)

Dataset	Majority Vote 50%					Majority Vote 100%				
	$+$	$-$	$?$	all	$-/+$	$+$	$-$	$?$	all	$-/+$
A	1297	649	54	2000	0.500	931	330	739	2000	0.354
B	1346	648	6	2000	0.481	846	307	847	2000	0.363
$avg.$	1321.5	648.5	30.0	2000	0.490	888.5	318.5	793.0	2000	0.359
C	1276	537	187	2000	0.421	616	194	1190	2000	0.315
D	188	274	79	541	1.457	28	96	417	541	3.429
E	1072	140	88	1300	0.131	588	35	677	1300	0.060

Table 3.4: Differences between majority votes 100% of A (experts) and B (students): pairs considered to be *disjoint* by the **students**.

A	B		
−	+	RailroadFacility	Pipeline
−	+	Order	Abstract
−	+	Newspaper	HomePage
−	+	School	MineSite
−	+	TelecomFacility	Monument
−	+	ReligiousLocation	Canal
−	+	InternationalOrganization	StockExchange
−	+	WaterRegion	PoliticalRegion
−	+	InternetDomain	EntitySource
−	+	ReligiousOrganization	Airline
−	+	RecreationalFacility	Capital
−	+	City	Archipelago
−	+	Pipeline	LaunchFacility
−	+	AstronomicalObject	Mountain
−	+	GovernmentOrganization	AmusementPark
−	+	AmusementPark	Galaxy
−	+	LaunchFacility	Bridge

1. the label and comment of a class often do not provide an unambiguous idea of what is meant with this class

2. some disjointness axioms may depend on the context: whereas *Dog* and *Livestock* may be disjoint in most parts of Europe, in the Chinese Wordnet[5] the latter is actually a hypernym of the former.

3. classes that have abstract individuals, like *Money*, *Message* or *Idea*

4. often the extension of two classes may be disjoint, although their intension is not, e.g. *US President* and *Woman*. Annotators struggle with this difference.

5. also, the extensions of two classes may be not disjoint, even though their intensions are: although *Weapon* and *Pitchfork* are disjoint intensionally (in the literal sense), their extensions do not need to be

6. mixing roles and so called basic classes, e.g. the role *Professor* and the *Person* itself that plays the role, which may be defined disjoint (depending on how roles are modeled [KSKM06])

[5] `http://www.keenage.com/`

Table 3.5: Differences between the majority votes 100% of A (experts) and B (students): pairs considered to be *disjoint* by the **experts**.

A	B		
+	−	Canal	Harbor
+	−	OfficialPoliticalMeeting	Parliament
+	−	Week	Month
+	−	Mountain	Peninsula
+	−	Island	Valley
+	−	Government	Parliament
+	−	Service	Telecom
+	−	Park	Festival
+	−	OilField	Province
+	−	Patent	AirplaneModel
+	−	Ministry	Location
+	−	Delta	River
+	−	TVCompany	Movie

7. mixing mereological and instantiation relations: a *Week* is part of a *Month*, so are these two classes disjoint? What about *Delta* and *River*?

8. mixing other types of relations with instantiation relations: see for example the pairs *Movie*/*TVCompany*, *Government*/*Parliament*, or *Patent*/*AirplaneModel*, where the instances have close relations and thus seem to confuse the annotators

9. instantiations at different levels of abstraction. E.g., when describing animals, *Eagle* may be the label of both an individual (e.g. of the class *Species*) and of a class itself. Are then the two classes *Species* and *Eagle* disjoint? (note that the individual *Eagle* is not the same as the class *Eagle*, but they may be connected via an axiom like $\textit{Class:Eagle} \equiv \exists \textit{species}.\{\textit{Individual:Eagle}\}$)

10. mixing lexical information with ontological one. The PROTON ontology contains concepts like *Alias* that form lexical information. Is a *JobTitle* disjoint from a *Job* or the *Person* having the *Job* or *JobTitle*?

Note that this list does not speak about problems of disjointness with regards to its definition in description logics, but rather with the problems our annotators had when they had to decide if two classes are disjoint or not. Many of the above problem types have a well-defined answer with regards to the formal semantics of disjointness, e.g. #7, where *Week* and *Month* are disjoint as they don't have common instances (since a week consists of seven days, and months consist of around 28-31 days. Note that the definition of week and month can change, but this basically means that we introduce new concepts who may or may not have the same name).

Recognizing the problem type would allow an ontology development environment to offer much more appropriate help than just a general description of the meaning of the disjointness axiom, which can be hard to apply at times.

Often the decision, if two classes are disjoint or not, will uncover underspecified or ambiguous classes, i.e. moot points in the description of one or both classes. Instead of simply adding (or, which is far harder to tract, *not* adding) a disjointness axiom, the rationale behind this decision should also be documented, following an ontology lifecycle methodology like DILIGENT [PTS04] for the continuous evolution and refinement of the ontology.

3.4 Evaluation: Learning a Classifier

In this section we present the evaluation procedure and analyse the results of the comparison between the classifier which has been trained on the features described in Section 3.2 and the sets of manual annotations (see Section 3.3).

3.4.1 Experimental Settings

To train the classifier we skipped pairs of classes tagged with $?$ since the definition of disjointness only distinguishes between disjoint and not disjoint classes. For the rest of the evaluation we will consider this two-class problem. We evaluate our learned classifier against two baseline: the random and majority baseline.

Random Baseline: The idea of the random baseline is to randomly choose the target class of the classifier. As we have a two-class problem we will distribute the pairs equally over the two classes. This will result in a 50% baseline for accuracy as 50% of the $+$ examples will be classified in $+$ which means that these examples are classified correctly. The same holds for the $-$ class.

Majority Baseline: The majority baseline is determined by taking the largest class as default classification. This way, we will get a high accuracy if the classes are unequally distributed. In this case, of course, the majority baseline is much more difficult to beat than the random baseline. Nevertheless, since in the experiments at hand we only have to deal with two classes ($+$ or $-$) which are not equally distributed, the majority baseline should be considered as more realistic than the random baseline.

Classifier settings: In order to be able to classify each pair of classes as being disjoint ($+$) or not ($-$), we trained a classifier based on the manual taggings created by human annotators (see section 3.3.2). The features for the classifier are the confidence values obtained from various sources as described in section 3.2.

We tested a couple of different classifiers made available by the Weka package[6]. In

[6] `http://www.cs.waikato.ac.nz/ml/weka/`

general, decision trees outperformed all other classifiers – maybe, because of the highly selective character of our features – while the performance of different types of decision trees was more the less comparable. Therefore, we finally chose the *ADTree* classifier [FM99] with default settings for our experiments which shows very good performance while at the same time providing interpretable results.

First, we performed a 10-fold cross-validation against the majority votes 100% and 50% of the datasets A (ontologists), B (students), C (all) and E (random) (cf. table 3.1a). The results for the random dataset are included to show the performance of our approach for an unbiased dataset (E contains examples chosen randomly from the set of all possible pairs without any selection criteria). To get the results for dataset D (siblings), we split dataset C into two independent parts - one for evaluation and one for training. The training set for the evaluation with dataset D consists of all manually tagged pairs except for the siblings.

3.4.2 Results

Table 3.7 and 3.6 list the results of our evaluation experiments by means of Precision (P), Recall (R), F-Measure (F) and Accuracy (Acc) (for definitions cf. [WF05]). From the tables it becomes evident that we easily beat the baselines for the datasets A (experts), B (students) and C in both cases majority vote 50% and 100%. With an accuracy of over 90% the performance of our system for dataset C is remarkable, especially in the case of the total majority vote. These results are comparable with the human inter-annotator agreement for experts and students – and even better for dataset C (90.9%) in comparison to the human agreement of 86.4%.

Dataset D, which only contains pairs of siblings, is certainly the most difficult to handle – for the classifier, but also for the human annotators – because, as explained in Section 3.3.3 siblings are semantically close, so that differences between their intensions and extensions may often be hard to grasp. As dataset D shows a relatively low average agreement compared to the other datasets (cf. table 3.2b)) the classifier seems to have more difficulties to learn it. This is also expressed by the very bad classification accuracy with 37% for majority vote 100%.

An investigation of the learned classifier revealed that the very important taxonomic feature (see Section 3.2.2) is not well populated in the siblings part of the dataset. To analyse the influence of this feature we constructed a dataset without this feature. As expected the accuracy for the training dataset drops, whereas for the evaluation set it is improved considerably from 37.9% to 74.2%. Moreover, the results for the majority vote 50% rise to 76.6% which can be interpreted as an indication to the noise insert by this feature.

Our approach seems to work very well also for the random dataset E as we got a better accuracy in both cases. The difference to the majority baseline is much smaller than for A, B, and C but the baseline of around 90% is very difficult to beat. To conclude, the

results – not only for the random dataset – are very promising and allow us to setup a competitive classifier to support ontology engineering.

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 [Qui93]. The ranking produced for data set C clearly indicates an exceptionally good performance of the features taxonomic overlap (Section 3.2.1), similarity based on WordNet and lexical context (Section 3.2.3), and Del.icio.us (Section 3.2.1). The contribution of other features such as the one presented in Section 3.2.4 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.

3.5 Related Work

Several ontology learning frameworks have been designed and implemented in the last decade. The Mo'K workbench [BNC00], for instance, basically relies on unsupervised machine learning methods to induce concept hierarchies from text collections. In particular, the framework focuses on agglomerative clustering techniques and allows ontology engineers to easily experiment with different parameters. OntoLT [BOS03] is an ontology learning plug-in for the Protégé ontology editor. It is targeted at end users and heavily relies on linguistic analysis, i.e. it makes use of the internal structure of noun phrases to derive ontological knowledge from texts. JATKE[7] is a Protégé based unified platform for ontology learning which allows for inclusion of modules for ontology learning. The OntoLearn framework [NVCN04] mainly focuses on the problem of word sense disambiguation, i.e. of finding the correct sense of a word with respect to a general ontology or lexical database. TextToOnto [MS01] is a framework implementing a variety of algorithms for diverse ontology learning subtasks. In particular, it implements diverse relevance measures for term extraction, different algorithms for taxonomy construction as well as techniques for learning relations between concepts. The recent RelExt approach [SB05] focusses on the extraction of triples, i.e. classes connected by a relation. None of the mentioned approaches deals with disjointness.

3.6 Conclusion and Future Work

Learning of disjointness axioms is an intuitive and useful extension of existing ontology learning frameworks. We have motivated the need for richter ontologies which include disjointness axioms and presented an approach consisting of a number methods to extract expressive feature for learning disjointness from different sources of evidence. In

[7] `http://jatke.opendfki.de/`

a thorough evaluation our learning approach behaved competitive to human annotators. As a by-product we captured lessons learned from human annotators with respect to their difficulties when modeling disjointness axioms.

Future work includes a combination with ontology evaluation approaches for richly axiomatized ontologies such as [Sch05]. Moreover, we want to integrate the novel methods into the Text2Onto [CV05a] framework for ontology learning from texts.

Acknowledgments: Research reported in this paper has been partially financed by the EU in the IST project SEKT (IST-2003-506826) (`http://www.sekt-project.com`). We would like to thank our colleagues, especially Peter Haase, for fruitful discussions, and all the students and colleagues at the AIFB and the University of Kassel for providing us with more than 10,000 taggings.

Table 3.6: Evaluation against Majority Vote 50% (ADTree)

Dataset	P			R			F			Acc	Acc_{random}	$Acc_{majority}$
	$+$	$-$	avg.	$+$	$-$	avg.	$+$	$-$	avg.			
A	0.815	0.638	0.727	0.823	0.626	0.725	0.819	0.632	0.726	0.757	0.500	0.666
B	0.807	0.642	0.725	0.844	0.580	0.712	0.825	0.609	0.717	0.758	0.500	0.675
$avg.$	0.811	0.640	0.726	0.834	0.603	0.719	0.822	0.621	0.722	0.758	0.500	0.671
C	0.854	0.682	0.768	0.874	0.644	0.759	0.864	0.663	0.764	0.806	0.500	0.704
D	0.558	0.628	0.593	0.255	0.861	0.558	0.350	0.726	0.538	0.615	0.500	0.593
E	0.910	0.761	0.836	0.990	0.250	0.620	0.948	0.376	0.662	0.904	0.500	0.884

Table 3.7: Evaluation against Majority Vote 100% (ADTree)

Dataset	P			R			F			Acc	Acc_{random}	$Acc_{majority}$
	$+$	$-$	avg.	$+$	$-$	avg.	$+$	$-$	avg.			
A	0.896	0.720	0.808	0.903	0.703	0.803	0.899	0.712	0.806	0.851	0.500	0.738
B	0.866	0.790	0.828	0.942	0.599	0.771	0.903	0.681	0.792	0.851	0.500	0.734
$avg.$	0.881	0.755	0.818	0.923	0.651	0.787	0.901	0.697	0.799	0.851	0.500	0.736
C	0.934	0.823	0.879	0.946	0.789	0.868	0.940	0.805	0.873	0.909	0.500	0.760
D	0.237	0.806	0.522	0.786	0.260	0.523	0.364	0.394	0.379	0.379	0.500	0.774
E	0.977	0.955	0.966	0.998	0.600	0.799	0.987	0.737	0.862	0.976	0.500	0.944

Chapter 4

Conclusion and Future Work

The work presented in this deliverable adds to the state-of-the-art in ontology learning by implementing new methods for ontology acquisition from Spanish resources, and the automatic generation of disjointness axioms.

For the future we will aim at a tighter integration of ontology learning, reasoning and evaluation by investigating possible applications of learning disjointness to ontology debugging, and the integration of reasoning into the learning process. We believe that a combination of semi-automatic means for ontology learning with different ontology evaluation measures and techniques for inconsistency detection and resolution will greatly support the process of engineering expressive ontologies in future ontology engineering environments.

4.1 Acknowledgements

Thanks to Sergi Fernández Langa his help in implementing the Spanish ontology learning algorithms – and Núria Casellas for performing the evaluation of Text2Onto in the context of the Legal Case Study.

Appendix A

Installation of Text2Onto (Spanish)

A.1 License and Availability

Text2Onto is published under the GNU Lesser General Public License (LGPL). Sources and binaries can be obtained from `http://ontoware.org/projects/text2onto/`.

A.2 Software Requirements

- Java 1.5+
- Any Java compatible operating system.
- GATE version 3.1
- English WordNet version 2.0
- Spanish WordNet (license required)
- TreeTagger and Spanish parameters

A.3 Installation (Windows)

1. Download Text2Onto from `http://ontoware.org/projects/text2onto/`.
2. Unzip file to <`T2O-DIR`> (e.g. `c:\text2onto`).

3. Install GATE 3.1[1] to <`GATE-DIR`>.
4. Copy the TreeTagger[2] (including the Spanish parameter file) to <`TT-DIR`>.
5. Install WordNet 2.0[3] to <`WN-DIR`>.
6. Copy Spanish WordNet[4] to <`T2O-DIR`>`3rdparty\spanishwordnet`.
7. Please note that none of the above mentioned directories should have a name which includes space characters, since this causes problems with the current version of the Text2Onto installer.
8. Make sure that you have installed the latest version of the Java virtual machine[5], since Text2Onto does not work with Java versions prior to 1.5.
9. Run the Text2Onto installer by executing `installer.bat` or <`T2O-DIR`>`installer\installer\myInstall.jar`. During the installation procedure you will be asked to specify the home directories of GATE, WordNet and the TreeTagger.
10. Start `text2onto.bat`.

A.4 Graphical User Interface

The graphical user interface of Text2Onto is composed of different views for the configuration of the ontology learning process and the presentation of the results (cf. figure A.1).

On the top left (A) there is a controller view, which can be used to set up an workflow by selecting appropriate algorithms for the different ontology learning tasks. The user may choose among a number of pre-defined strategies for combining the results of these algorithms (see figure A.2).

In the bottom left corner (B) the user will find a corpus view, which allows him to set up a corpus by specifying the text documents the ontology should be extracted from.

The panel on the right (C) shows the results of the current ontology learning process. There are different tabs - one for each type of modeling primitives extracted from the corpus.

---

[1] `http://gate.ac.uk`
[2] `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/`
[3] `http://wordnet.princeton.edu/`
[4] `http://www.lsi.upc.edu/~nlp/web/index.php`
[5] `http://java.sun.com`

And finally, below the results panel there is a view for debugging output and GUI error messages. Please note that most of the run-time information generated by Text2Onto is still printed to the command line.

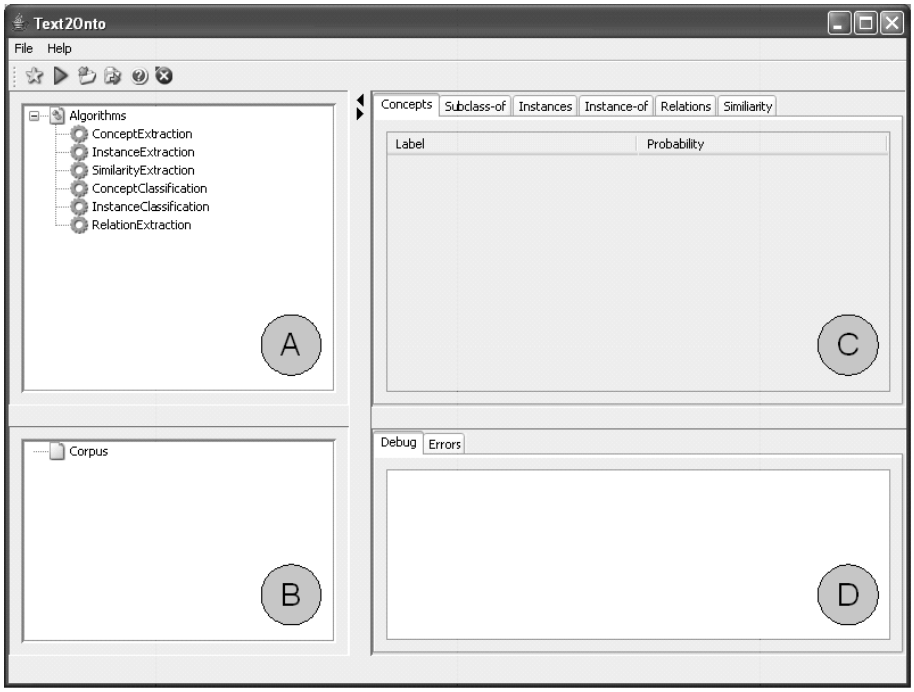

Figure A.1: GUI

In order to start the ontology learning process, the user can select *File→ Run* from the main menu or just press the appropriate toolbar button.

Moreover, the *File* also allows to start a new ontology learning session (*New*), import an existing ontology (*Import*), export the POM to a concrete KAON or RDFS ontology (*Export*) and exit Text2Onto (*Exit*).

Once, an ontology has been extracted from the corpus the different modeling primitives are displayed to the user, who can interact with the POM by giving feedback to individual learning results (cf. figure A.3).

A maximum degree of traceability is given by the fact that the user can not only view the change history of any ontology element, but also get a natural language explanation for all modeling decisions of the system (see figure A.4).

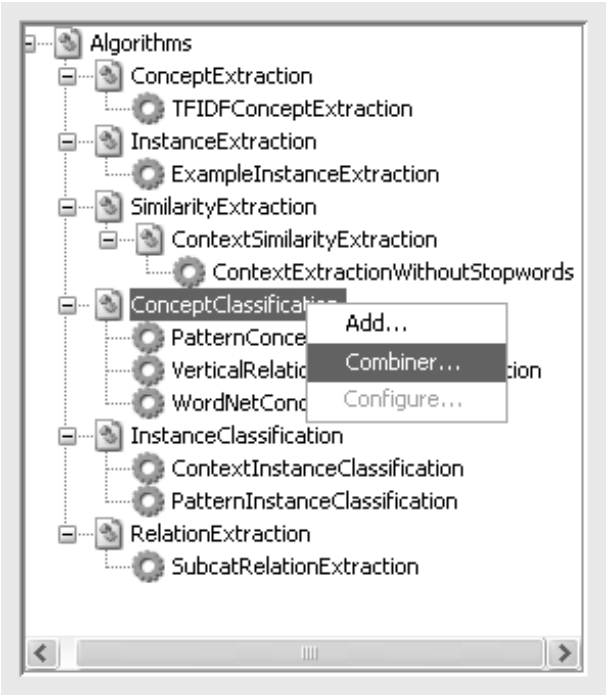

Figure A.2: Controller View

A.5 API

In addition to the graphical user interface, Text2Onto features a java-based API which provides users and developers with programmatic access to the complete functionality of the ontology learning framework. This programming interface allows for integrating Text2Onto in other software applications and facilitates the development of new ontology learning algorithms.

The following example (cf. listing A.1) shows how to set up a simple ontology learning workflow including one type of concept extraction and different concept classification algorithms for learning subclass-of relationships. The resulting POM is then transformed into an OWL ontology.

Listing A.1: API

```
Corpus corpus = CorpusFactory.newCorpus( sCorpusDir );
POM pom = POMFactory.newPOM();
AlgorithmController ac =
      new AlgorithmController( corpus, pom );

// concept extraction
ac.addAlgorithm( new TFIDFConceptExtraction() );

// concept classification
ComplexAlgorithm conceptClassification =
      new ComplexAlgorithm();
conceptClassification.setCombiner( new AverageCombiner() );
ac.addAlgorithm( conceptClassification );

ac.addAlgorithmTo( conceptClassification,
      new PatternConceptClassification() );
ac.addAlgorithmTo( conceptClassification,
      new VerticalRelationsConceptClassification() );
ac.addAlgorithmTo( conceptClassification,
      new WordNetConceptClassification() );

ac.execute();

OntologyWriter writer = new OWLWriter( pom );
writer.write( new URI( "pom.owl" ) );
```

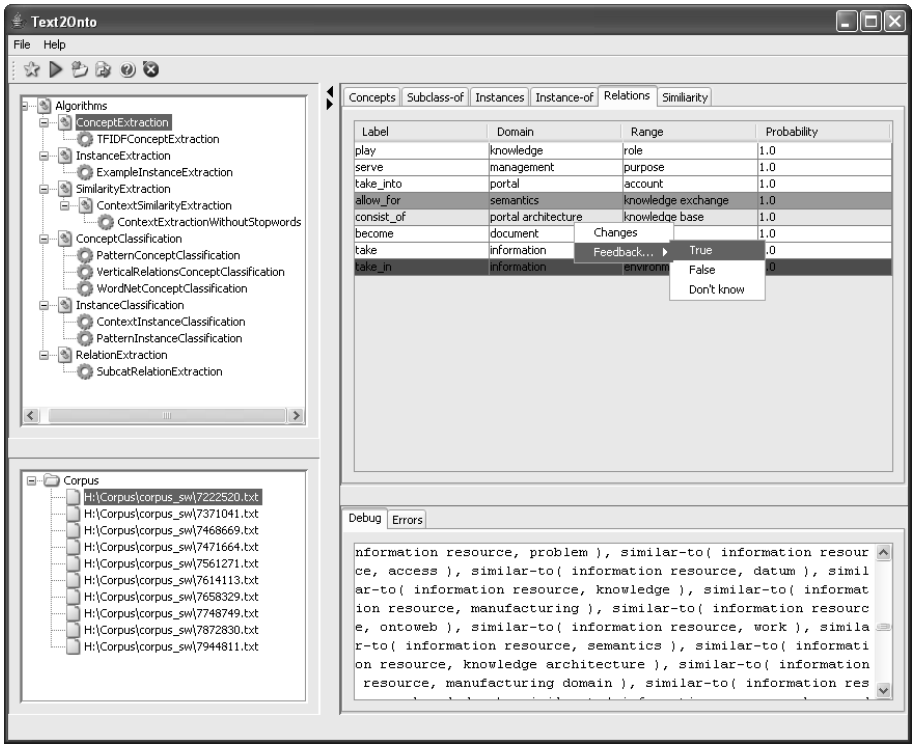

Figure A.3: User Feedback

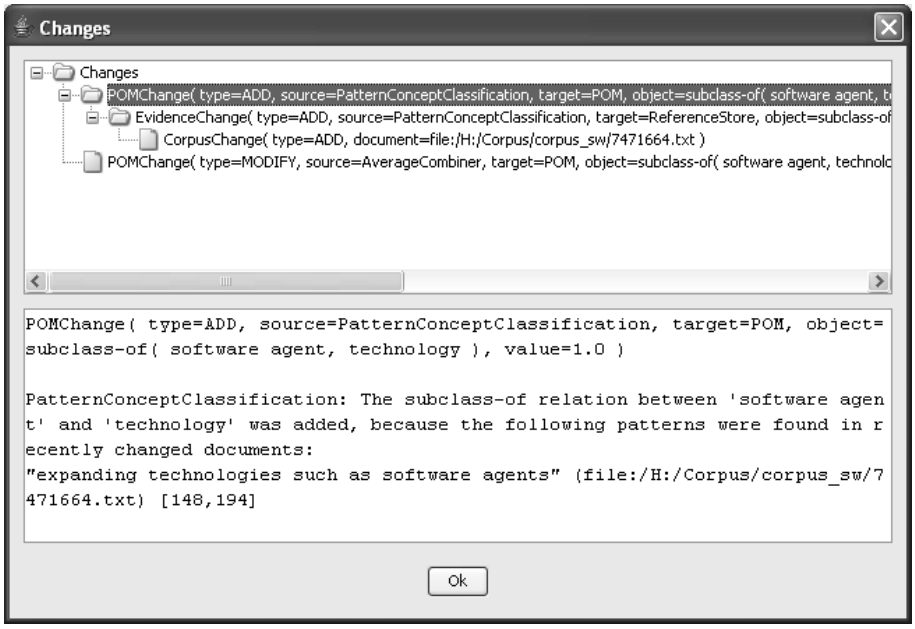

Figure A.4: Changes

Bibliography

[BNC00] G. Bisson, C. Nedellec, and L. Canamero. Designing clustering methods for ontology building - The Mo'K workbench. In *Proceedings of the ECAI Ontology Learning Workshop*, pages 13–19, 2000.

[BOS03] P. Buitelaar, D. Olejnik, and M. Sintek. OntoLT: A protégé plug-in for ontology extraction from text. In *Proceedings of the International Semantic Web Conference (ISWC)*, 2003.

[BTS06] E. Paslaru Bontas, C. Tempich, and Y. Sure. ONTOCOM: A cost estimation model for ontology engineering. In I. Cruz et al., editors, *Proceedings of the 5th International Semantic Web Conference (ISWC 2006)*, volume 4273 of *LNCS*, pages 625–639. Springer-Verlag Berlin Heidelberg, 2006.

[CLS05] Philipp Cimiano, Günter Ladwig, and Steffen Staab. Gimme the context: Context-driven automatic semantic annotation with c-pankow. In Allan Ellis and Tatsuya Hagino, editors, *Proceedings of the 14th World Wide Web Conference*, pages 332 – 341, Chiba, Japan, MAY 2005. ACM Press.

[CMBT02] 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 Annual Meeting of the ACL*, 2002.

[CV05a] P. Cimiano and J. Völker. Text2onto – a framework for ontology learning and data-driven change discovery. In *Proc. of the 10th Int.l Conf. on Applications of Natural Language to Information Systems (NLDB'05)*, June 2005.

[CV05b] 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)*, pages 166–172, Borovets, Bulgaria, September 2005. INCOMA Ltd.

[CV05c] Philipp Cimiano and Johanna Völker. Towards large-scale, open-domain and ontology-based named entity classification. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP'05)*, pages 166–172, SEP 2005.

[FM99] Yoav Freund and Llew Mason. The alternating decision tree learning algorithm. In *ICML*, pages 124–133, 1999.

[GW00] N. Guarino and C. A. Welty. A formal ontology of properties. In *Knowledge Acquisition, Modeling and Management*, pages 97–112, 2000.

[Har68] Z. Harris. *Mathematical Structures of Language*. Wiley, 1968.

[HV05] Peter Haase and Johanna Völker. Ontology learning and reasoning - dealing with uncertainty and inconsistency. In *Proc. of the Workshop on Uncertainty Reasoning for the Semantic Web (URSW)*, pages 45–55, 2005.

[KSKM06] K. Kozaki, E. Sunagawa, Y. Kitamura, and R. Mizoguchi. Fundamental considerations of role concepts for ontology evaluation. In *Proc. of the Workshop EON – Evaluation of Ontologies for the Web*, 2006.

[Mil95] G. Miller. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[MS01] A. Maedche and S. Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2), 2001.

[NVCN04] R. Navigli, P. Velardi, A. Cucchiarelli, and F. Neri. Extending and enriching WordNet with OntoLearn. In *Proc. of the GWC 2004*, pages 279–284, 2004.

[PC06] N. Casellas M. Poblet M. Blázquez J. Contreras V.R. Benjamins J.-M. López Cobo P. Casanovas, J.-J. Vallbé. D10.4.1 after analysis. 2006. with the collaboration of Z. Huang, and J. Völker.

[PTS04] H. Sofia Pinto, Christoph Tempich, and Steffen Staab. Diligent: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engingeering of ontologies. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI)*, 2004.

[Qui93] J. R. Quinlan. *C4.5 Programs for Machine Learning*. Morgan Kaufmann, California, 1993.

[RDH+04] A. Rector, N. Drummond, M. Horridge, J. Rogers, H. Knublauch, R. Stevens, H. Wang, and C. Wroe. OWL pizzas: Practical experience of teaching OWL-DL – common errors & common patterns. In *Proc. of EKAW 2004*, pages 63–81, 2004.

[RSW02] T.G. Rose, M. Stevenson, and M. Whitehead. The reuters corpus volume 1-from yesterdays news to tomorrows language resources. *Proc. of the Third International Conference on Language Resources and Evaluation*, pages 29–31, 2002.

[SB05] A. Schutz and P. Buitelaar. RelExt: A tool for relation extraction in ontology extension. In *Proc. of the 4th International Semantic Web Conference (ISWC2005)*, 2005.

[Sch05] S. Schlobach. Debugging and semantic clarification by pinpointing. In *Proc. of the 2nd European Semantic Web Conference (ESWC2005)*, volume 3532 of *LNCS*, pages 226–240. Springer, 2005.

[TKM04] I. Terziev, A. Kiryakov, and D. Manov. Base upper-level ontology (BULO) guidance. SEKT deliverable 1.8.1, Ontotext Lab, Sirma AI EAD (Ltd.), 2004.

[VS05a] J. Völker and Y. Sure. Data-driven change discovery. SEKT deliverable 3.3.1, Institute AIFB, University of Karlsruhe, 2005.

[VS05b] J. Völker and Y. Sure. Data-driven change discovery. evaluation. SEKT deliverable 3.3.2, Institute AIFB, University of Karlsruhe, 2005.

[VVS05] J. Völker, D. Vrandečić, and Y. Sure. Automatic evaluation of ontologies (AEON). In *Proc. of the 4th International Semantic Web Conference (ISWC2005)*, volume 3729 of *LNCS*, pages 716–731. Springer, 2005.

[Wan06] Taowei David Wang. Gauging ontologies and schemas by numbers. In *Proc. of the Workshop EON – Evaluation of Ontologies for the Web*, 2006.

[WF05] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Sys. Morgan Kaufmann, 2nd edition, June 2005.

[WP94] Zhibiao Wu and Martha Palmer. Verb semantics and lexical selection. In *32nd. Annual Meeting of the Ass. for Computational Linguistics*, pages 133–138, New Mexico, 1994.