D3.3.2 Data-driven Change Discovery - Evaluation

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

Abstract.
EU-IST Integrated Project (IP) IST-2003-506826 SEKT
Deliverable D3.3.2 (WP3.3)
In this deliverable we present the evaluation of our work performed in the task ‘T3.3 Data-driven Change Discovery’.
Keyword list: Data-driven Change Discovery, Ontology Learning, Ontology Management

Document Id.	SEKT/2005/D3.3.2/v1.0
Project	SEKT EU-IST-2003-506826
Date	December 28, 2005
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

In this deliverable, we present and evaluate two approaches for discovering changes in ontologies based on domain data analysis and formal ontology evaluation.

The first approach relies on the ontology learning framework Text2Onto in order to extract different types of ontology elements from a changing corpus. Since all ontology learning tasks are executed in an incremental manner, potential modifications to the ontology will be proposed whenever the corpus changes. A detailed evaluation of the ontology learning results showed very promising results, but also revealed a number of problems related to the evaluation procedure and individual user perception. In particular, we found a high disagreement among the human annotators especially with respect to the identification and the classification of instances. Apparently, an appropriate judgment of the learned ontology elements is at least as difficult as the task of ontology learning itself.

The second approach towards data-driven change discovery applies the OntoClean methodology in order to detect formal errors in a given taxonomy. For this purpose we developed AEON, the first and only framework for the automatic formal evaluation of ontologies. The evaluation of AEON yielded very good results, which could even be improved by the use of a larger training set and some parameter tuning of the classifiers.

Text2Onto and AEON can be seen as two complementary approaches being implemented independently of each other. Both rely on a linguistic analysis of natural language text in order to suggest possible changes in ontologies. Whereas Text2Onto has been developed to discover any kind of changes (including the addition of new concepts, instances or relations), the main purpose of AEON is to detect formal errors in the taxonomic hierarchy and to identify subclass-of relationships which have to be removed or modified in order to restore the formal correctness of the ontology.

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 [VS05] the ontology learning framework Text2Onto has been designed to provide an infrastructure for data-driven change discovery in the SEKT project. The aim of this deliverable is to refine the analysis of possible application scenarios for Text2Onto and to evaluate the results of the ontology learning algorithms.

1.3 Application Scenario

For the purpose of this deliverable we decided to focus on the BT Digital Library case study. Since a much more detailed description of the case study has already been given by [HS05] this chapter draws only a rough outline of this scenario focussing on aspects relevant for the current evaluation tasks.

The main purpose of Digital Library case study is to investigate the use of Semantic Web technologies for searching, browsing and maintaining BT's Digital Library. The library offers its users a single interface to different databases containing both PDF and

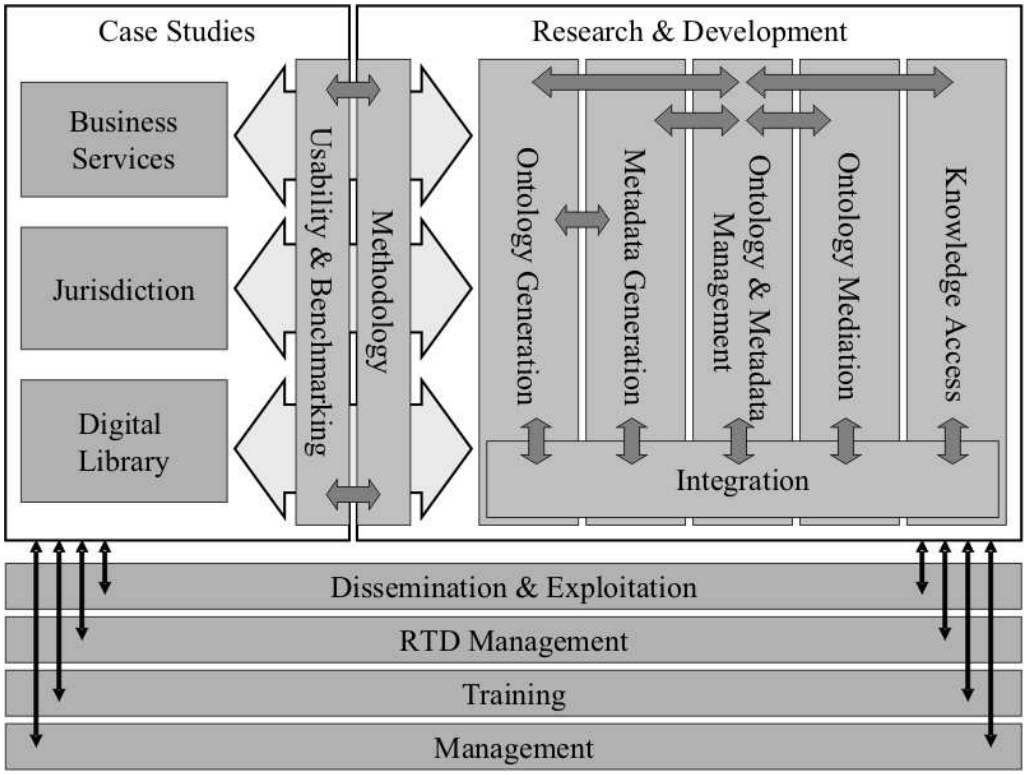

Figure 1.1: The SEKT Big Picture

HTML documents from a variety of publishers. All databases are regularly updated by adding new documents and corresponding metadata. In order to support the users in keeping track of those contents which are related to their personal interests and in finding new potentially interesting documents they can create or register for so-called information spaces. Each information space consists of a number of documents associated with one or more topics defined by a global topic hierarchy. The contents of each information space vary from plain text to semi-structured PDF or HTML documents.

1.3.1 Fine-grained Topic Hierarchies

One use case we would like to address in the context of the Digital Library case study is the one of searching and browsing via fine-grained topic hierarchies describing the contents of each information space. Such a topic ontology could be automatically generated and updated by Text2Onto or any other automatic or semi-automatic ontology learning environment. Other than the overall Digital Library topic ontology, a topic hierarchy learned from an individual information space, would contain very fine-grained topics which specifically characterize the content of that particular information space. Once each document in the information space has been classified according to one or more top-

ics in this fine-grained hierarchy this would allow for (i) enhanced searching and browsing (document and sentence level) and (ii) better visualization of the information space. Moreover, (iii) based on usage statistics with respect to the learned ontology topics within the fine-grained topic hierarchy could be proposed for integration into the global topic ontology.

1.3.2 Expressive Domain Ontologies

While the goal of topic learning is restricted to learning fairly light-weight ontologies, here the challenge is to extract complex ontologies from the text corpora, exploiting more of the expressiveness of the OWL ontology language [SWM04]. This in turn allows evaluating complex, structured queries against the knowledge extracted from the documents. A possible query (expressed in natural language) might be: ”Find all companies specializing in genomic technology in America.” Please note that the focus here is not so much on retrieving documents, but on answering questions (while of course it is also possible to ask for the documents from which the knowledge has been extracted). To be able to support such expressive queries, an appropriate user interface will be required.

Whereas ontology learning will be handled by Text2Onto, the query processing is going to be performed by KAON2. In addition, the consistent ontology evolution functionality guarantees the consistency of the knowledge during the incremental learning, and thus ensures the meaningfulness of query answering.

1.3.3 Incremental Ontology Updates

Since the content of each information space changes at least once a week when new documents are added to the Digital Library the knowledge which is represented by such an information space evolves continuously over time. This means that the corresponding topic or domain ontologies will have to be updated regularly by adding new concepts, instances or relations. In order to do this in an efficient and (logically) consistent way the ontology learning algorithms used for generating the ontologies should be able to react to changes in the corpus and to adapt precisely those parts of the existing ontology which are affected by these changes. A first approach towards data-driven change discovery has been presented in D3.3.1 [VS05]. Please note that this task directly relates to the work performed in workpackage 3.1 dealing with consistent ontology evolution [HS05].

1.4 Overview

In the following we present and evaluate two approaches towards discovering changes in ontologies based on textual data. Chapter 2 describes the ontology learning framework Text2Onto and the different algorithms for extracting ontologies from natural language

texts (see section 2.2). An overview of the evaluation setting (section 2.3) is followed by a detailed analysis of the evaluation results (section 2.4). The second approach based on AEON, a tool for the automatic formal evaluation of ontologies, is presented in chapter 3. Section 3.2 describes the architecture of AEON. The evaluation setting and the results of our experiments are described in section 3.3 and 3.4 respectively. Finally, chapter 4 concludes with a summary and an outlook to future work.

Chapter 2

Change Discovery based on Domain Data Analysis

2.1 Introduction

The first approach we developed for discovering changes in ontologies is based on the ontology learning framework Text2Onto to extract various types of ontology elements from a given corpus of text documents. Since all ontology learning tasks are executed in an incremental manner, modifications to the ontology can be derived by considering changes to the corpus.

In this chapter we briefly summarize our previous work by describing the main features of Text2Onto. We further present the evaluation setting we used to measure the quality of the ontology learning results before finally concluding with a detailed analysis of our findings.

2.2 Approach

2.2.1 Text2Onto

Text2Onto is a framework for ontology learning and ontology evolution developed as part of our efforts in task 3.3 (cf. section 1). Since a detailed description of the architecture and the algorithms has already been given as part of deliverable D3.3.1 [VS05], this chapter presents only a brief summary of the main characteristics of Text2Onto.

A typical usage scenario for Text2Onto is depicted by figure 2.1. The user specifies a corpus, e.g. a collection of text, HTML or PDF documents, and starts the graphical workflow editor. The editor provides her with a list of algorithms which are available for the different ontology learning tasks, and assists her in setting up an appropriate workflow

for the kind of ontology she wants to learn as well as to customize the individual ontology learning algorithms to be applied. Once the ontology learning process is started, the corpus gets preprocessed by a natural language processing component (GATE [CMBT02]), before it is passed to the algorithm controller. In the following, depending on the configuration of the previously specified workflow, a sequence of ontology learning algorithms is applied to the corpus. Each algorithm starts by detecting changes in the corpus and updating the reference store accordingly. Finally, it returns a set of requests for changes regarding the POM, i.e. the *Preliminary Ontology Model* to its caller, which could be the algorithm controller, but also a more complex algorithm (cf. figure 2.2). After the process of ontology extraction is finished, the POM is presented to the user.

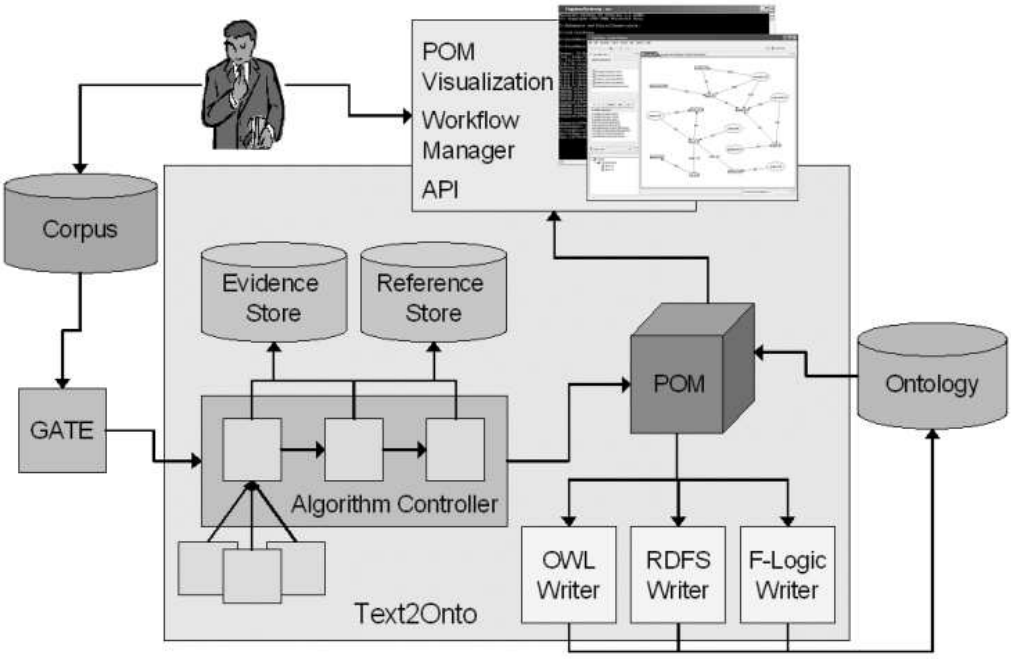

Figure 2.1: Text2Onto - Usage Scenario

Since the POM unlike any concrete ontology is able to maintain thousands of conflicting modeling alternatives in parallel, an appropriate and concise visualization of the POM is of crucial importance for not overwhelming the user with too much information. Although several pre-defined filters such as a probability threshold will be available for *pruning* the POM, some user interaction might still be needed for transforming the POM into a high-quality ontology. After having finished her interaction with the POM, e.g. after adding or removing concepts, instances or relations, the user can select among various ontology writers, which are provided for translating the POM into different ontology representation languages.

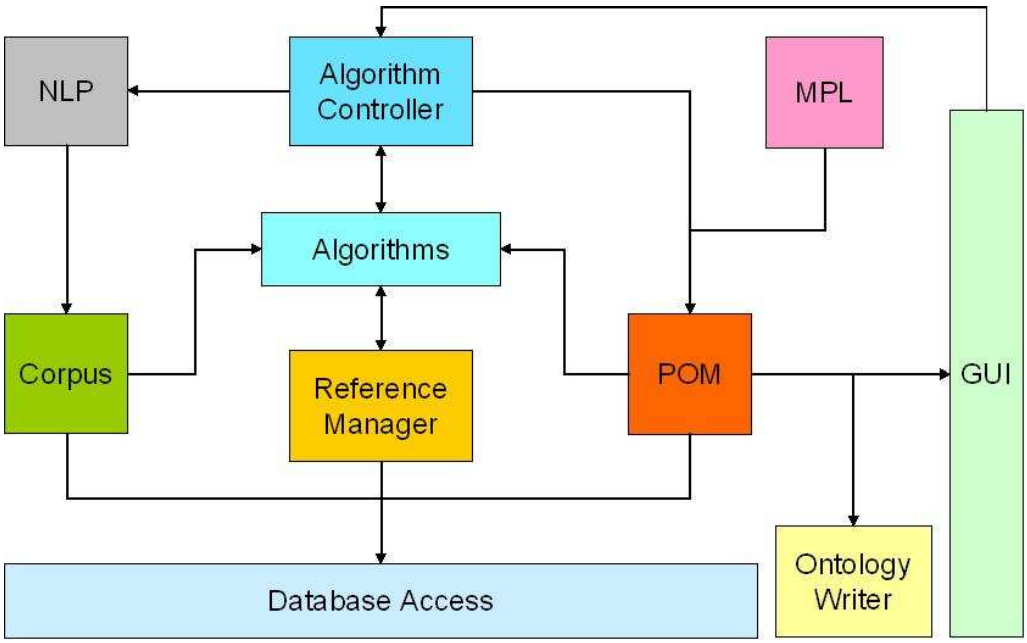

Figure 2.2: Text2Onto - Architecture

2.2.2 Methods

Concepts and Instances Different term weighting measures are used to compute the relevance of a certain concept or instance with respect to the corpus: Relative Term Frequency, TFIDF [Sal91], Entropy [Gra90] and the C-value/NC-value method in [KF98]. For each term, the values of these measures are normalized into the interval $[0..1]$ and used as corresponding relevance in the POM.

Subclass-of Relations In order to learn subclass-of relations, we have implemented a variety of different algorithms exploiting the hypernym structure of WordNet [Mil95], matching Hearst patterns [Hea92] in the corpus as well as in the WWW and applying linguistic heuristics mentioned in [VNCN05]. The resulting confidence values of these algorithms are then combined through combination strategies as described in [CPSTS05].

Instance-of Relations In order to assign instances or named entities appearing in the corpus to a concept in the ontology Text2Onto relies on a similarity-based approach extracting context vectors for instances and concepts from the text collection and assigning instances to the concept corresponding to the vector with the highest similarity with respect to their own vector [CV05]. Alternatively, we also implemented a pattern-matching algorithm similar to the one used for discovering part-of relations.

General Relations To learn general relations, Text2Onto employs a shallow parsing strategy to extract subcategorization frames (e.g. `hit(subj,obj,pp(with))`, transitive + PP-complement) enriched with information about the frequency of the terms

appearing as arguments [MS00]. These subcategorization frames are mapped to relations such as `hit`(*person*,*thing*) and `hit_with`(*person*,*object*). The confidence is estimated on the basis of the frequency of the subcategorization frame as well as of the frequency with which a certain term appears at the argument position.

For the purpose of discovering **part-of relations** in the corpus, we developed regular expressions matching lexico-syntactic patterns as described in [CB99] and implemented an algorithm counting the occurrences of patterns indicating a part-of relation between two terms t_1 and t_2, i.e. part-of(t_1,t_2). The confidence is then calculated by dividing by the sum of occurrences of patterns in which t_1 appears as a part. The results are combined with confidences which can be acquired by consulting WordNet for mereological relations.

Equivalence and Equality Following the assumption that terms are similar to the extent to which they share similar syntactic contexts, we implemented algorithms calculating the similarity between terms on the basis of contextual features extracted from the corpus, whereby the context of a terms varies from simple word windows to linguistic features extracted with a shallow parser. This corpus-based similarity is then taken as the confidence for the equivalence of the corresponding concepts or instances.

Disjointness For the extraction of disjointness axioms we implemented a simple heuristic based on lexico-syntactic patterns. In particular, given an enumeration of noun phrases $NP_1, NP_2, ...(and|or)NP_n$ we conclude that the concepts $C_1, C_2, ...C_k$ 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.

Subtopic-of Relations

In order to identify subtopic-of relationships, we implemented an approach by [SC99]. It is based on the assumption that each topic tends to occur in a true subset of all the documents containing its supertopic. Therefore, it creates subtopic-of relationships by considering inclusion relationships between the document sets associated with all concepts in the ontology. Additionally, we also developed a very simple algorithm that generates subtopic-of relation from previously extracted subclass-of relationships.

2.3 Evaluation

2.3.1 Data

To evaluate the quality of the ontology learning results we decided to select 1,700[1] from the 4,196 abstracts in the 'Knowledge Management' information space. Since each of these abstracts summarizes the major ideas of the corresponding document and includes the most important concepts with respect to the document's topic, this decision enables us to achieve a very good coverage of the information space content while still avoiding performance problems which would arise when processing the complete collection of full-text documents.

2.3.2 Implementation

For the ontology extraction process we developed concrete implementations of the methods described in section 2.2.2. The names of the corresponding algorithms are given below.

Concepts

- `TFIDFConceptExtraction` extracts the most important concepts with respect to a given domain (e.g. *knowledge* for the domain of *knowledge management*). The relevance of each concept is determined by computing its average $TF * IDF$ value with respect to the whole document collection.

Instances

- `TFIDFInstanceExtraction` extracts the most important instances with respect to a given domain (e.g. *OWL* for the domain of *Semantic Web*). The relevance of each instance is determined by computing its average $TF * IDF$ value with respect to the whole document collection.

Subclass-of Relations

- `PatternConceptClassification` is based on a predefined set of domain-independent patterns.
- `VerticalRelationsConceptClassification` applies a simple heuristic.
- `WordNetConceptClassification` uses WordNet [Mil95] to obtain evidence for subclass-of relationships.

[1] For technical reasons we had to restrict the corpus to 1,000 abstracts when learning subtopic-of relationships.

Instance-of Relations

- `PatternInstanceClassification` is based on a predefined set of domain-independent patterns.

Non-taxonomic Relations

- `SubcatRelationExtraction` extracts non-taxonomic relationship by considering the arguments of verbs which are found in the corpus.

Disjointness

- `EnumerationDisjointExtraction` is based on the assumption that terms which are part of an enumeration often represent disjoint concepts.

Subtopic-of Relations

- `SubtopicOfRelationExtraction` considers the occurrences of concepts in the corpus. It assumes that a subtopic tends to occur in a subset of those documents which are associated with its supertopic.
- `SubtopicOfRelationConversion` creates subtopic-of relationships by considering taxonomic and non-taxonomic relationships previously extracted from the corpus.

In the case where more than one algorithm is given for a certain task (such as the extraction of subclass-of relations, for example) an *average combiner* was used to combine the confidence values generated by the individual algorithms. Each time a certain ontology element has been extracted simultaneously by different algorithms (e.g. `subclass-of(researcher, person)` by `PatternConceptClassification` and `WordNetConceptClassification`) this kind of combiner updates the confidence value for that particular element by averaging the confidence values computed by all algorithms.

A typical ontology learning workflow starts by the extraction of concept and instances which are particularly relevant for the domain of interest. Subsequently, subclass-of and optionally non-taxonomic relationships among the concepts will be extracted from the corpus. Instance-of relations are learned to populate the resulting taxonomy with the instances previously extracted. This workflow can be extended by algorithms for the extraction of disjointness axioms, if a more expressive ontology is required. Or, if the user is interested in a subtopic hierarchy, methods for learning subtopic-of relationships can be included in the workflow.

2.3.3 Measures

An important assumption when trying to measure the quality of the ontology learning results is that the domain of interest can be appropriately modeled by the given corpus of text documents - in this case, randomly selected from the (*knowledge management*) information space. We also assume that all human experts revert to a certain amount of background knowledge when judging the quality of individual ontology elements, because in practice, a certain amount of domain expertise will required for semi-automatic or computer-aided ontology engineering with Text2Onto. Based on these assumptions we performed a detailed evaluation of Text2Onto in the following way:

First, the top 50 results of each ontology learning subtask (e.g. the subclass-of relationships with the highest confidence values) were given to five human annotators[2] considered to be familiar with the domain of 'knowledge management'. All of them had to judge the correctness or relevance of the extracted concepts, instances and relations. To each of these ontological primitives they assigned a score between 1 (totally incorrect) to 4 (perfectly correct) according to the following scale.

For example, if Text2Onto suggests a taxonomic relation such as `subclass-of(internet, entertainment)` you might say that this is 'somehow correct'. Although a classification of `internet` as some kind of `network` would be more appropriate, it is very often used as a medium for leisure and entertainment.

Additional examples are given in parenthesis.

concept

- 1 - no entity (`always`)
- 2 - no concept, i.e. more likely to be an instance (`John Davis`)
- 3 - correct, but irrelevant for the domain (`number`)
- 4 - correct and relevant (`knowledge management`)

instance

- 1 - no entity (`why`)
- 2 - no instance, i.e. more likely to be a concept (`work`)
- 3 - correct, but irrelevant for the domain (`GMT`)

[2]We decided not to do the evaluation automatically, because it would have caused too many problems to build a gold standard ontology for the corpus described in section 2.3.1. Moreover, it is much easier for humans to identify subtle differences in meaning and to distinguish between different degrees of correctness as described below

- 4 - correct and relevant (`UML`)

subclass-of

- 1 - totally incorrect (`subclass-of(software, person)`)
- 2 - somehow correct (`subclass-of(database, application)`)
- 3 - domain/range too general/specific (`subclass-of(software developer, living being)`)
- 4 - perfectly correct (`subclass-of(computer, electronic device)`)

instance-of

- 1 - totally incorrect (`instance-of(John Davis, application)`)
- 2 - somehow correct (`instance-of(York Sure, professor)`)
- 3 - domain/range too general/specific (`instance-of(Peter Haase, female)`)
- 4 - perfectly correct (`instance-of(OWL, ontology representation language)`)

subtopic-of

- 1 - totally incorrect (`subtopic-of(knowledge, nature)`)
- 2 - somehow correct (`number, algorithm`)
- 3 - domain/range too general/specific (`subtopic-of(molecular biology, science)`)
- 4 - perfectly correct (`subtopic-of(relational databases, databases)`)

non-taxonomic relation

- 1 - totally incorrect (`know(hardware, intelligence)`)
- 2 - somehow correct (`write(software, information)`)
- 3 - domain/range too general/specific (`produce(entity, product)`)
- 4 - perfectly correct (`find(person, information)`)

disjointness

- 1 - totally incorrect (`disjoint(programming, software development)`)
- 2 - small overlap (`disjoint(country, city)`)
- 3 - too general/specific (`disjoint(knowledge, image processing software)`)
- 4 - perfectly correct (`disjoint(software, hardware)`)

Finally, for each type of ontology element (e.g. the learned subclass-of or instance-of relationships) we computed the average scores for all human annotators. In order to obtain a single measure for the performance of Text2Onto in the regarding ontology learning subtask we further calculated the average over all individual user ratings.

2.4 Results

In this section we present the results of our ontology learning experiments with Text2Onto. Each of the following diagrams shows the average ratings of the five annotators with respect to the top 50 results for the regarding ontology element (see 2.3). The right-most bar of each diagram, which represents the average over all of the five annotation sets, can be considered as an indicator for the overall performance of Text2Onto in the regarding ontology learning subtask.

2.4.1 Concepts and Instances

Both concepts and instances presented to the human annotators were ranked according to their estimated relevance for the domain of interest, i.e. *knowledge management*. Possible ratings were: $1 =$'no entity', $2 =$'no concept (instance)', $3 =$'correct, but not relevant for the domain' and $4 =$'correct and relevant'.

As shown by the first diagram 2.3 the average user rating for the extracted concepts is pretty high, close to the maximum value of 4. That means, almost all of the extracted concepts were considered to be correct and reasonable concepts and also relevant for the domain in question.

Surprisingly, the results for the extracted instances are much worse. The diagram depicted by figure 2.4 shows that the overall average value for the instances is not more than about 2.5. One possible explanation could be, that at least one of the annotators assigned a rating of 1 to each instance which he did not know and therefore could not recognize as a valid instance (e.g. `CVW` which is indeed a collaboration software environment).

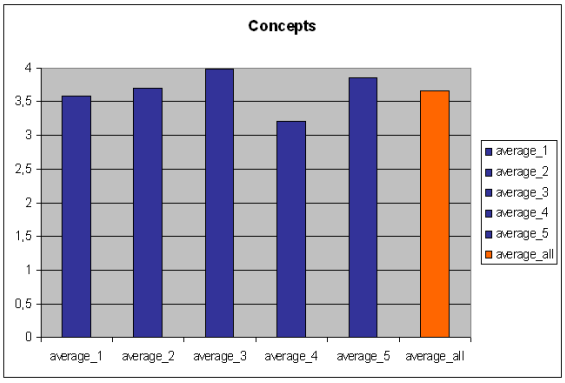

Figure 2.3: Average User Rating - Concepts

It is important to mention that the evaluation was made under the assumption that all annotators were completely familiar with the domain of *knowledge management*. Although at least some of them had access to the corpus, they were not explicitly told to use it for the verifying the results. Finally, most of them used Google or any other search engine to look up unknown instances.

Obviously, one possible improvement for future evaluations of Text2Onto might be to provide the annotators with the corresponding text fragments for each of the learned ontology elements. Moreover, the annotators should be give the possibility to mark some of the results as 'unknown', so that they do not feel forced to assign a rating even in those cases where they are unsure about the meaning of the regarding ontology elements. This is especially important in case of a corpus which contains many abbreviations that tend to be highly ambiguous.

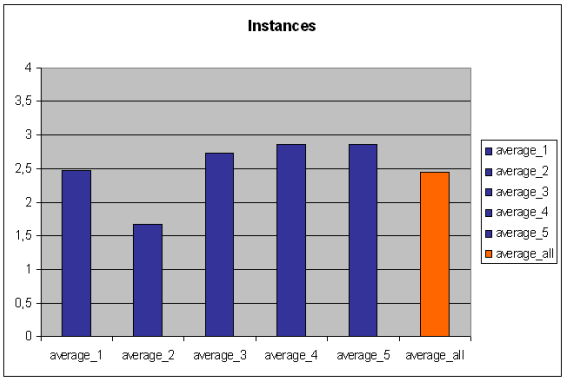

Figure 2.4: Average User Rating - Instances

2.4.2 Subclass-of Relations

Figure 2.5 presents the evaluation results for subclass-of relationships. The average user rating ranges from 1.5 to 2.22, which means that there is a relatively high agreement between the five annotators. An overall average of approximately 1.9 is a bit worse than we expected, but certainly a solid basis for future work.

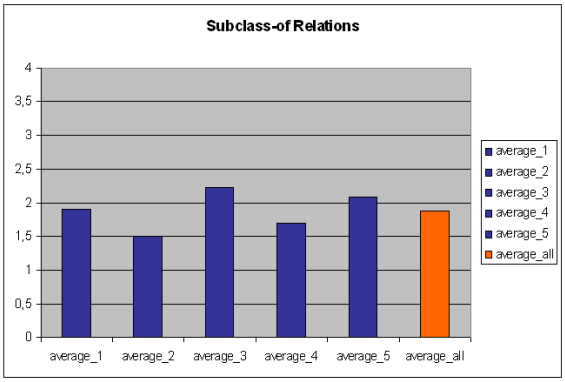

Figure 2.5: Average User Rating - Subclass-of Relations

2.4.3 Instance-of Relations

Whereas the human agreement regarding the average judgement of the subclass-of relationships is relatively high (see above), the average user rating for the instance-of relations varies enormously from annotator to annotator. As shown by diagram 2.6 two of the annotators assigned an average value of only about 1.5 to the extracted relationships; two others delivered an average judgement of approximately 2.7.

The reasons for this are certainly similar to those we already observed when considering the evaluation of the instances. In many cases the instances or abbreviations involved in a particular instance-of relation were unknown to the annotator who then assigned a very low rating to this relationship.

2.4.4 Subtopic-of Relations

As depicted by figure 2.7 the results we got by evaluating the extraction of subtopic-of relations are comparable to those obtained for the subclass-of relationships (see above). The main difference consists in a higher difference between the average user ratings. In particular, those annotators who assigned rather high average ratings to the subclass-of relations assigned even better ratings to the subtopic-of relationships, whereas those annotators who considered the subclass-of relations to be rather bad were even less content with the subtopic-of relationships.

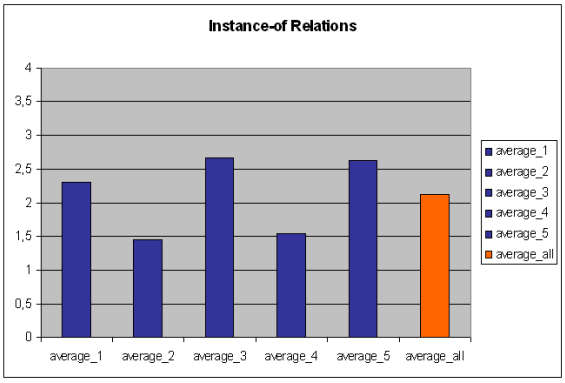

Figure 2.6: Average User Rating - Instance-of Relations

This phenomenon might partially be explained by the fact that many of the subtopic-of relations were derived from subclass-of relationships (cf. section 2.2). Users who can imagine arbitrary domain relevant concepts to be topics in a certain application scenario will be perfectly agreed with this kind of conversion. But those users who are very critical with respect to the formal correctness of any kind of taxonomic relationships will certainly find that the entities involved in a particular subtopic-of relation might eventually be considered as concepts, but not as topics in a narrower sense.

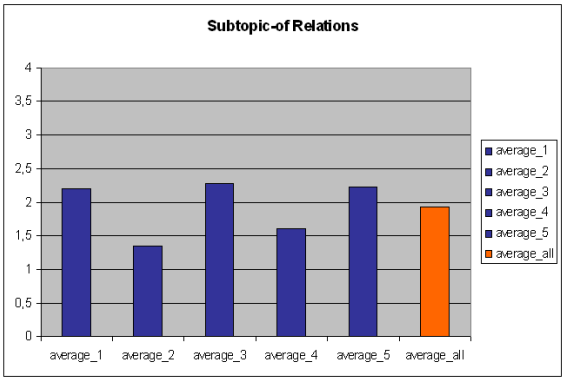

Figure 2.7: Average User Rating - Subtopic-of Relations

2.4.5 Disjointness Axioms

Given that the approach we used for the extraction of disjointness axioms is based on a very simple heuristic (see section 2.2) the evaluation of the learned axioms yielded surprisingly good results. As shown by diagram 2.8 the overall average rating is about 2.1 and there seems to be a pretty high agreement among the human annotators about the quality of the results.

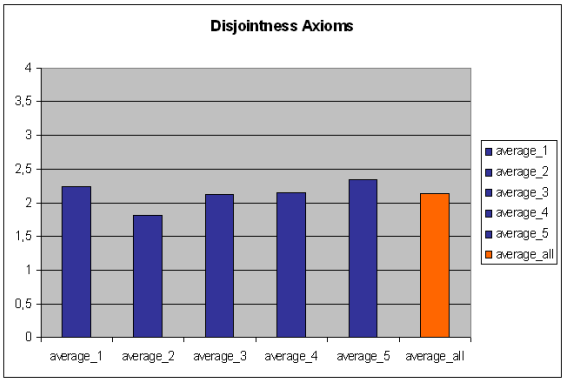

Figure 2.8: Average User Rating - Disjointness Axioms

2.4.6 Non-taxonomic Relations

When considering figure 2.9 which presents the evaluation results for the extraction of non-taxonomic relationships you can observe an extremely high difference among the average user ratings. Again it was the second annotator who assigned a very low score of only about 1.4 to the extracted relationships, whereas the other annotators assigned average ratings between 2.1 and approximately 3.3.

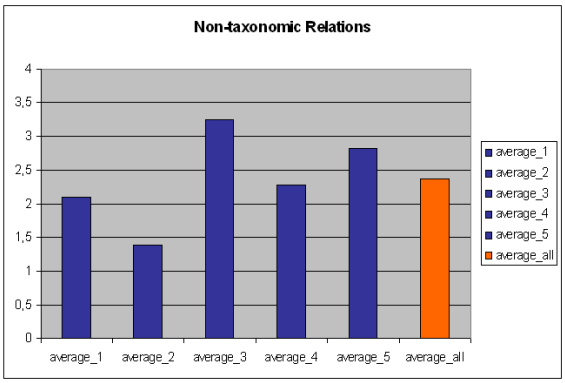

Figure 2.9: Average User Rating - Non-taxonomic Relations

Based on these promising results we are now aiming at a detailed technical evaluation of Text2Onto within the case studies with particular focus on the incremental ontology learning techniques. We will develop a number of change strategies in order to customize the change discovery process with respect to different users and application scenarios (cf. chapter 4) and evaluate our approach with a larger number of users.

Chapter 3

Change Discovery based on Ontology Evaluation

3.1 Introduction

In addition to the change discovery by domain data analysis (see chapter 2) we developed an approach to suggest ontology changes based on formal ontology evaluation. In particular, our approach uses OntoClean - a well-known and still the only methodology for formal ontology evaluation - to automatically identify formal errors in taxonomic hierarchies. The main motivation for us to choose OntoClean among the huge variety of ontology evaluation methods comes from the requirements of the application scenario (see chapter 1). Since we are aiming at the generation of expressive domain ontologies for question answering (cf. section 1.3.2), which relies to a large extend on ontology reasoning, the formal correctness of the learned ontologies will be of major importance for the quality of the results.

We implemented our approach in terms of AEON [VVS05] which is intended to constitute a flexible framework for the *A*utomatic *E*valuation of *ON*tologies. The next sections provide a brief overview of OntoClean and the technical implementation of AEON, followed by a detailed description of the evaluation settings and results.

3.2 Approach

3.2.1 OntoClean

We provide a brief introduction to OntoClean, for a more thorough description refer to [GW00], for example. In the OntoClean vocabulary, *properties* are what is commonly called *concepts* or *classes*. *Meta-properties* are therefore properties of properties. Within

this paper we will use the term *meta-property* in the usual OntoClean way, whereas we will refrain from using the term *property* but rather stick to the more common term *concept*. OntoClean consists of two steps: first every single concept needs to be tagged with occurrences of the core meta-properties, which are described below. Thus, every concept will have a certain tagging like *+R+U-D+I*. We call an ontology with tagged concepts a tagged ontology (wrt. OntoClean, to be precise). After the tagging, the second step of OntoClean is to check all subsumption relations of the ontology (also called Subclass-relations). OntoClean constraints the possible taxonomic relations by disallowing subsumption relations between specific combinations of tagged concepts. This way, OntoClean provides a unique approach by formally analyzing the concepts intensional content and their subsumption relationships.

We now briefly present the four main meta-properties and rules which belong to OntoClean. The four meta-properties are: *Rigidity (R)*, *Unity (U)*, *Dependence (D)* and *Identity (I)*. They base on philosophical notions dating back to Aristotle. Here we will offer a short description of these meta-properties.

Rigidity. Rigidity is based on the notion of *essence*. A concept is essential for an instance *iff*[1] it is necessarily an instance of this concept, in all worlds and at all times. *Iff* a concept is essential to all of its instances, the concept is called rigid and is tagged with *+R*. *Iff* it is not essential to some instances, it is called non-rigid, tagged with *-R*. An anti-rigid concept is one that is not essential to all of its instances. It is tagged $\sim$*R*. An example of an anti-rigid concept would be *teacher*, as no teacher has always been, nor is necessarily, a teacher, whereas *human* is a rigid concept because all humans are necessarily humans and neither became nor can stop being a human at some time.

Unity. Unity is about "What is part of something and what is not?" This answer is given by an **Unity Criterion** (UC), which is true for all parts of an instance of this concept, and for nothing else. For example, there is an unity criterion for the parts of a human body, as we can say for every human body which parts belong to it. Concepts carrying an UC have Unity and are tagged *+U* else *-U*.

Dependence. A concept C_1 is dependent on a concept C_2 (and thus tagged *+D*), *iff* for every instance of C_1 an instance of C_2 must exist. An example for a dependent concept would be *food*, as instances of food can only exist if there is something for which these instances are food. Another way to regard dependency is to distinguish between intrinsic and extrinsic concepts. Intrinsic concepts are independent, whereas extrinsic concepts need to be given to an instance by circumstances or definitions.

Identity. A concept with Identity is one, where the instances can be identified as being the same at any time and in any world, by virtue of this concept. This means that the concept carries an **Identity Criterion** (IC). It is tagged with *+I*, and with *-I* otherwise. It is not important to answer the question of what this IC is (this may be hard to answer), it is sufficient to know that the concept carries an IC. For example, the concept *human*

[1] if and only if (`http://en.wikipedia.org/wiki/If_and_only_if`)

carries an IC, as we are able to identify someone as being the same or not, even though we may not be able to say what IC we actually used for that. On the other hand, a concept like *red* would be tagged *-I*, as we cannot tell instances of red apart because of its color.

On a tagged ontology, we can use the existing OntoClean rules to check the ontology for consistency. Here, we will give only one illustrative example for these rules. For a full list refer to [GW04]. As shown in [SAS03] such rules can be formalized as logical axioms and validated by an inference engine.

$\sim R$ can't subsume $+R$. Having a concept *C* subsuming the concept *D*, with *C* tagged $\sim R$ and *D* tagged $+R$, would lead to the following inconsistency: *D* must always hold true for all of its instances. *D*, as a subsumed concept, would always imply *C* for all of its instances. Therefore there are at least some instances of *C* that are necessarily *C* as they are *D*. Thus *C* can not be anti-rigid, as the tagging says, because this would mean that it is not necessarily true for any of its instances – which would be a contradiction. The classic example is *student*, an anti-rigid concept, subsuming *human*, a rigid concept, which is obviously wrong: whereas every student is free to leave the university and stop being a student, humans cannot stop being humans. As every human would be a student, according to the example, they never could stop being a student, which contradicts the previous sentence.

3.2.2 Implementation

Our approach for the automatic assignment of meta-properties according to the OntoClean methodology is based on three fundamental assumptions. First, we believe that the nature of concepts is to some degree reflected by human language and what is said about instances of these concepts in the language corpus. Because of this, we consider statistics about the occurrences of lexico-syntactic patterns (see section 3.2.2) as a feasible means to capture the meta-properties of ontological concepts. Second, in line with similar approaches by [Gre99], [KLO02], [RS03], [CHS04] and [CLS05] we think that using the Web as a corpus is an effective way of addressing the typical data sparseness problem one encounters when working with natural language corpora. Finally, from our point of view, the Web being the biggest source of common-sense knowledge available constitutes a perfect basis for computational comprehension of human intuition as to the philosophical notions of essence, unity and identity.

Architecture

In order to evaluate our approach we developed AEON, a tool which matches lexico-syntactic patterns on the Web to obtain positive and negative evidence for rigidity, unity, dependence and identity of concepts in an RDFS or OWL ontology. The architecture of AEON is roughly depicted by figure 3.1. It consists of an *evaluation component*, which is responsible for training and evaluation, a *classifier* for mapping given sets of evidence

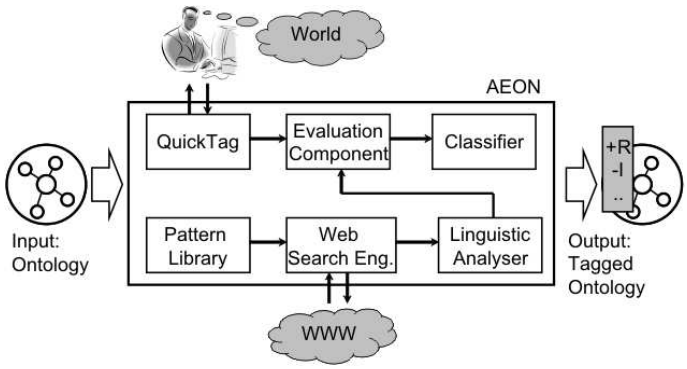

Figure 3.1: Architecture of AEON

to meta-properties such as +R or -U, a *pattern library* and a *search engine wrapper*.

The **pattern library** is initialized by means of an XML file containing a set of abstract patterns for each meta-property (see listing 3.1). Each of these patterns includes a specification of the type of evidence it produces, e.g. negative evidence for rigidity. Moreover, it contains a declaration of one or more variables and a set of Web queries which can be instantiated by replacing the regarding variables by the labels of the concepts to be analysed. Finally, a linguistic filter, i.e. a regular expression over tokens and part-of-speech tags, is defined for filtering the results obtained by the above mentioned queries (see section 3.2.3).

Listing 3.1: Negative Evidence for Rigidity (R)

```
<pattern>
  <variable name="x" />
  <evidence type="false" for="R" />
  <google regex="is\t\w+ no\t\w+ longer\t(DT\w+\t)?(NN|NP|
     NNS|NPS) x\t[^(NN|NP|NNS|NPS)]">
    <query string="is no longer a x" />
    <query string="is no longer an x" />
    <query string="is no longer x" />
  </google>
</pattern>
```

Given a set of instantiated patterns (e.g. *"is no longer a student"*) the **search engine wrapper** uses the Google™API in order to retrieve web pages or snippets, i.e. parts of web pages containing the regarding search string, from the Web. For normalization purposes (see below) it also queries the web for all occurrences of the regarding concept, such as *"student"* for example.

The **linguistic analyser** provides methods for tokenization, lemmatizing and part-of-

speech (POS) tagging[2], which are required for some fundamental preprocessing of the snippets and HTML pages obtained from the Web and for an appropriate matching of the linguistic patterns described above. By what we call *Linguistic Filtering* we analyse, for example, all those snippets returned by Google™, which satisfy the query "is no longer a computer" (cf. listing 3.1). If the regular expression associated with the query does not match, the particular snippet is not counted as a hit and thus does not provide any evidence with respect to the rigidity of `computer`. This way, we avoid false matches in case of statements such as "He is no longer a computer hacker." or (this would yield false evidence for the unity of `employee`) when we find a phrase like "the computer of an employee consists of". Of course, linguistic filtering is also applied in the normalization process (see above).

Finally, for each pattern i contained in the above mentioned pattern library the positive or negative evidence $evidence(p, i, c)$ for a concept c having a certain meta-property $p \in \{R, U, D, I\}$ is given by:

$$evidence(p, i, c) = \frac{\sum_{q \in Q_i} lf(hits(q_c))}{lf(hits(c))},$$

where Q_i is the set of queries associated with pattern i, q_c is the instantiation of query q for concept c, and $hits(q_c)$ and $hits(c)$ are the number of hits obtained for q_c or c respectively. lf is a function implementing the linguistic filtering described above.

Given a concept c and the evidence values obtained for all patterns the decision whether or not a meta-property p applies to c is made by a classifier. A set of classifiers – one for each meta-property – has been trained on a small number of examples provided by human annotators (cf. section 3.3). The manual effort rests with the creating of a gold standard ontology and classifiers to be trained on this ontology.

Patterns

During the last decades, lexico-syntactic patterns have become generally accepted as an effective means for extracting various types of lexical and ontological relationships such as hyponymy and meronymy (cf. [Hea92], [CB99], [HS98]). Nevertheless, there has been little if any work on the use of pattern-based approaches towards the extraction of meta-properties, i.e. properties of concepts or relations. So, we performed an extensive evaluation of many different pattern candidates before finally choosing a small subset of particularly promising patterns for the evaluation of our approach. All of these patterns are domain-independent, thus being well suited for the WWW as a very heterogeneous corpus.

[2]For part-of-speech tagging we used QTag, a probabilistic java-based tagger which is available for non-commercial use (`http://www.english.bham.ac.uk/staff/omason/software/qtag.html`).

Rigidity. The intuition behind the patterns we defined for Rigidity is the following: If any individual can become or stop being a member of a certain class, then it holds that the membership of this class, e.g. the property *being a student*, is not essential for all its individuals. Therefore, we can obtain **negative** evidence with respect to Rigidity from the following patterns:

`is no longer (a|an)?` *`CONCEPT`*

`became (a|an)?` *`CONCEPT`*

`while being (a|an)?` *`CONCEPT`*

Unity. As explained in section 3.2.1 a concept is tagged with $+U$ if for each of its instances all parts can be identified and if they share a common Unity Criterion which holds true for exactly these parts. Because of this, in order to determine whether a given concept has unity or not we have to find answers to questions such as *"what is part of an object? and what is not?"* or *"under which conditions is the object a whole?"*. If we can answer these questions for at least most of the instances of the concept, we can take this as **positive** evidence for Unity.

`part of (a|an)?` *`CONCEPT`*

Moreover, since instances of concepts which are not countable usually do not carry a unity criterion, we can get **positive** evidence for Unity by searching for the following patterns:

`(one|two)` *`CONCEPT`*

Of course, `one` and `two` seem to be somewhat arbitrary, but since Google™is not yet able to process queries containing regular expressions we had to confine ourselves to what we considered as the most frequent of all possible variations of this pattern.

Similarly, **negative** evidence can be obtained by a pattern which indicates non-countability of a concept.

`amount of` *`CONCEPT`*

Identity. According to [GW04] identity is given by the fact that two instances of a concept are the same *iff* they have the same parts. This is known as *mereological extensionality* and can be expressed by the following patterns providing **positive** evidence for Identity:

`CONCEPT` `consists of (two|three) parts`

`CONCEPT` `is composed of (two|three) parts`

Additional **positive** evidence for identity can be obtained by the rather straightforward pattern:

CONCEPT is identified by

Negative and **positive** evidence respectively can be obtained by these merely linguistic patterns checking whether the name of the concept is an adjective or a noun.

Both patterns are matched on the results of Google™queried for nothing but the concept name. Please note that linguistic preprocessing as described in section 3.2.2 is required to allow this kind of lexico-syntactic pattern matching, since these patterns assume the text to be an alternate sequence of words and POS tags. The tags *JJ*, *JJR* and *JJS* indicate an adjective, whereas *NN*, *NP*, *NNS* and *NPS* are indicators for a common or proper noun.

(JJ|JJR|JJS) *CONCEPT*

(NN|NP|NNS|NPS) *CONCEPT*

Also, countability means that the instances of a concept are obviously identifiable (or else they would not be countable). Therefore we reuse the same patterns that we have already used as positive or negative evidence for Unity.

(one|two) *CONCEPT*

amount of *CONCEPT*

Dependence. Among the meta-properties Rigidity, Unity, Identity and Dependence we consider Dependence as the most difficult one to learn automatically. Maybe, this is because of the fact that relational knowledge, i.e. knowledge involving more than one concept, is required in order to detect Dependence. Nevertheless, we tried to capture Dependence of concepts by the following pattern:

cannot be (a|an)? *CONCEPT* without

Additional Patterns. Due to the flexible architecture of AEON, adding further patterns is a very easy task. It simply requires the addition of the pattern in described format to the XML file.

We had some more patterns in mind, but preliminary testing in Google™ revealed often only a small number of hits, which would only lower the efficiency of the system and not improve the output of the system adequately.

3.2.3 Discussion

The described approach is original, and quite a number of problems were raised. We solved many of them, but some remain for further research. Both kinds are described in this section.

Certain patterns could return a lot of inappropriate evidence. Searching for the frag-

ment *"is no longer a computer"* would also return *"is no longer a computer hacker"*, which is false evidence about the Rigidity of computers. To solve this problem we introduced linguistic preprocessing and patterns that recognize *computer* not being the subject of the given example. Thus we can get rid of a lot of false evidence.

The other problem occurs with high level, abstract or seldom used concepts: they just do not return hits, or return only a small, and thus usually unreliable number of evidence. However, we do not consider this as a big problem in general, since this kind of very abstract concepts mostly appear in upper-level ontologies which are typically smaller and less dynamic than domain ontologies. If we do not get any hits, the concept will not be part of possible constraint errors. So it does not really bother the user with wrong warnings but rather simply ignores this concept.

A much bigger problem is given by the highly ambiguous nature of human language. So far, our approach does not distinguish between different concepts which could be meant by the word "glass", for example. Whereas the "glass" which can be used to drink water certainly has Unity, the "glass" windows are made of does not have Unity. Linguistic patterns do not help in this case. We will try to solve this problem by comparing the context of the word – given by a Google™snippet or a Web page – with the semantic neighborhood of the regarding concept.

Natural language is not as strict and formal as the OntoClean meta-properties. The best known example is the English verb *to be*, which can have various meanings based heavily on context, like subsumption, definition or constitution. But exactly these different meanings play a crucial role within the OntoClean methodology. Thus, the translation of the OntoClean definitions of meta-properties to commonly used language patterns was quite challenging. With the patterns given in this section we hope to have achieved a good balance between language ambiguity, pragmatic indication of meta-properties and number of occurrences for a wide range of concepts.

The combination of negative and positive evidence right now just happens by simple subtraction. Maybe more complex combinations will yield even better results. This is an open issue. So is the difference between Non-, Anti- and Semi-Rigidity. Right now we just consider Rigidity and Non-Rigidity, but the more detailed division may lead to an even better evaluation of the ontology.

3.3 Evaluation

3.3.1 Data

As described in section 3.3.4 we decided to use the System, Top and Upper module of the Proton ontology[3] for the evaluation of our approach. The merged ontology consists

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

of 266 concepts, most of them annotated with a short natural language description. The list of all concepts together with their descriptions was given to three human annotators in the following called A_1, A_2 and A_3. All of them were considered to be experts in using the OntoClean methodology. Nevertheless, whereas Rigidity, Identity and Dependence were considered by all annotators, only two of them also assigned Unity labels to some of the concepts. Table 3.1 shows the number of concepts and their corresponding taggings created by each of the human annotators. The data sets labeled A_1/A_2, A_1/A_3, A_2/A_3 were obtained by building the intersection of two of the single data sets. Obviously, $A_1/A_2/A_3$, which is the intersection of all three data sets – the set of concepts which are tagged identically by all human annotators – is extremely sparse.

In order to illustrate how difficult it was for the human annotators to tag the ontology according to the OntoClean methodology we measured the human agreement between the data sets. *strict* means that two taggings were considered equal only if they were totally identical. *relaxed* means that $-$ and $\sim$ were considered the same. Since our approach so far does not distinguish between Semi- and Anti-Rigidity, for example, the strict agreement can be neglected for the following evaluation. As shown by table 3.2 the average human agreement is extremely low, which means close to the random baseline and sometimes much lower than the results we obtained by automatic tagging. Given these figures indicating the difficulty of this task, we believe any kind of automatic support could be of great use for formal ontology evaluation.

3.3.2 Baseline

In order to obtain an objective baseline for the evaluation of AEON which is statistically more meaningful than the human agreement (see table 3.2) we computed a random baseline for the F-Measure as follows: Let x be the overall number of concepts to be tagged, p the number of positive and $n = x - p$ the number of negative examples. Given a random tagging for all n concepts we can assume that half of them are tagged as $+$ and how many are tagged as $-$. Of course, the fraction of positives within the whole data set tends to be the same as in each of the randomly chosen subsets S_+ and S_- of size $\frac{n}{2}$. Therefore, the number of true positives (TP) and true negatives (TN) is given by $TP = \frac{p}{x} * \frac{x}{2} = \frac{p}{2}$ and $FP = (1-\frac{p}{x})*\frac{x}{2} = \frac{x}{2}-\frac{p}{2} = \frac{x-p}{2} = \frac{n}{2}$ whereas the false positives (FP) and false negatives (FN) can be computed by $TN = \frac{n}{x} * \frac{x}{2} = \frac{n}{2}$ and $FN = (1 - \frac{n}{x}) * \frac{x}{2} = \frac{x}{2} - \frac{n}{2} = \frac{x-n}{2} = \frac{p}{2}$.

Obviously, the Precision P_+ for the positive examples (for example, all concepts tagged as *+R*) is given by $P_+ = TP/(TP + FP)$, whereas the Precision for the negative examples can be obtained by $P_- = TN/(TN + FN)$. Recall can be computed by $R_+ = TP/(TP + FN)$ and $R_- = TN/(TN + FP)$ respectively.

Given Recall and Precision we can obtain the F-Measure for positive and negative examples by $F_+ = \frac{2*P_+*R_+}{P_++R_+}$ and $F_- = \frac{2*P_-*R_-}{P_-+R_-}$. This leads to an *macro-average F-Measure* of $F = \frac{1}{2} * (F_+ + F_-)$, which we consider as a reasonable baseline for the evaluation of our approach. A detailed overview of the concrete baselines we determined

	R			U			I			D		
	+	-	∼	+	-	∼	+	-	∼	+	-	∼
A_1	147	69	50	156	81	29	194	61	11	151	110	3
A_2	208	39	0	103	138	3	189	58	0	31	203	13
A_3	201	64	0	0	0	0	223	42	0	63	1	0
avg	**185.3**	**57.3**	**16.7**	**86.3**	**73.0**	**10.7**	**202.0**	**53.7**	**3.7**	**81.7**	**104.7**	**5.3**
A_1 / A_2	122	3	20	77	61	11	134	17	4	23	94	3
A_1 / A_3	125	27	15	0	0	0	171	18	1	47	1	0
A_2 / A_3	161	14	0	0	0	0	163	12	0	9	0	0
avg	**136.0**	**14.7**	**11.7**	**25.7**	**20.3**	**3.7**	**156.0**	**15.7**	**1.7**	**26.3**	**31.7**	**1.0**
A_1 / A_2 / A_3	106	2	6	0	0	0	126	8	0	9	0	0

Table 3.1: Tagged Concepts

	A_1 / A_2		A_1 / A_3		A_2 / A_3		A_1 / A_2 / A_3	
	relaxed	strict	relaxed	strict	relaxed	strict	relaxed	strict
R	58.7%	50.6%	63.0%	57.4%	71.1%	71.1%	46.3%	43.9%
U	61.1%	56.6%	N/A	N/A	N/A	N/A	N/A	N/A
I	66.4%	64.8%	71.7%	71.3%	71.1%	71.1%	54.5%	54.5%
D	48.9%	45.7%	75.0%	75.0%	15.0%	15.0%	15.0%	15.0%
avg	**58.8%**	**54.2%**	**69.9%**	**67.9%**	**52.4%**	**52.4%**	**38.6%**	**37.8%**

Table 3.2: Human Agreement

for all data sets is given by table 3.3.

3.3.3 Setting

Setting. Since we decided to evaluate our system separately for *R*, *U*, *I* and *D*, we made 2*7*4=56 experiments (one for each human annotator, each meta-property, with and without linguistic filtering) using a number of Weka[4] classifiers. In order to detect the limitations of our approach and to see what we can potentially get out of the data we are able to provide, we first tried many different types of classifiers, such as Support Vector Machines, Bayesian classifiers and Decision Trees. Since the latter turned out to perform best we finally decided to focus on the class of Decision Trees – among them ADTree, RandomForest and J48, for example. The features given to these classifiers were sets of evidences obtained by all patterns for the regarding meta-property (see section 3.2.2). Precision, Recall and F-Measure for both positive and negative examples as well as the macro-average F-Measure were determined by a 10-fold cross-validation. Please note that for training and evaluation we only used those concepts which were annotated in the regarding data set and for which we obtained at least some evidence. The percentage of tests which failed, because we did not get any Google™hits for the instantiated patterns was about 20% for rigidity, 5% for identity and around 10% for unity. Because of this, in many cases the number of examples we gave to the classifiers was extremely low - especially for the agreement data sets A_1/A_2, A_1/A_3, A_2/A_3 and $A_1/A_2/A_3$. The reason why the results are nevertheless very promising, certainly is the good quality of the classification features we get by using a pattern-based approach.

3.3.4 Lessons Learned

For the evaluation and training of our automatic methods, we needed a gold standard tagging of an ontology with the OntoClean meta-properties. Although OntoClean is already some years old and appeared in a number of publications, actual tagged ontologies were

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

	R			U			I			D		
	+	-	M-avg	+	-	M-avg	+	-	M-avg	+	-	M-avg
A_1	52.5	47.2	49.9	54.0	45.3	49.6	59.3	35.1	47.2	53.5	45.9	49.7
A_2	62.7	24.0	43.4	45.8	53.6	49.7	60.5	32.0	46.2	20.1	63.6	41.8
A_3	60.1	32.6	46.4	N/A	N/A	N/A	62.7	24.1	43.4	66.3	3.0	34.7
avg	**58.4**	**34.6**	**46.6**	**49.9**	**49.5**	**49.7**	**60.8**	**30.4**	**45.6**	**46.6**	**37.5**	**42.1**
A_1 / A_2	62.7	24.1	43.4	50.8	49.1	50.0	63.6	20.4	42.0	27.7	61.8	44.7
A_1 / A_3	60.0	33.5	46.7	N/A	N/A	N/A	64.3	16.7	40.5	66.2	4.0	35.1
A_2 / A_3	64.8	13.8	39.3	N/A	N/A	N/A	65.1	12.1	38.6	66.7	N/A	N/A
avg	**62.5**	**23.8**	**43.1**	**50.8**	**49.1**	**50.0**	**64.3**	**16.4**	**40.4**	**53.5**	**32.9**	**39.9**
A_1 / A_2 / A_3	65.0	12.3	38.7	N/A	N/A	N/A	65.3	10.7	38.0	66.7	N/A	N/A

Table 3.3: Random Baseline (F-Measure)

found only extremely scarcely. Our best resource was the example ontology in [GW04] and some examples in the other publications. This amounted to about 30-40 tagged concepts. [WMCC04] describes the creation of another ontology evaluated with OntoClean, but this is not publicly available. To the best of our knowledge there are no further available tagged ontologies.

So, we decided to tag an ontology on our own. We wanted a generic, domain-independent ontology with a not too small number of concepts. This is to ensure that the experience we gain and the classifiers trained will be most reusable for further ontologies evaluated with AEON in the future. We chose Proton[5] and merged the System, Top and Upper modules. The resulting ontology contained 266 concepts, as diverse as *Accident*, *Alias*, *Happening* or *Woman*.

We asked methodology and ontology engineering experts to tag Proton according to the OntoClean methodology, because we wanted to base the evaluation of our own techniques on this human tagging. Most of them told us that based on their experience with OntoClean the manual tagging of an ontology such as Proton would take more than one week. Some even considered this as an effort of one month – which would of course render any evaluation of the ontology far too expensive to be efficient. Finally, we were able to convince two of them to create a manual tagging of Proton. The third tagging we used for our evaluation was done by one of the authors of this deliverable.

The tagging itself was very strenuous, and often uncertainty arose. Decisions were debatable and the documentation of OntoClean was open to interpretation. The experts tagged the ontology in the given time of four to six hours, but they achieved an agreement far lower than expected (refer to table 3.2). Concepts similar to those in the example ontology in [GW04] were often tagged consistently, but the agreement on the other concepts was low (close to the baseline given by random tagging). This suggests that the experts rather worked by analogies (not surprisingly, given the time constraints) to the examples (an approach that is very common for humans) than by applying the definitions of the meta-properties.

Taking into account that OntoClean is only a method to evaluate the taxonomic relationships of an ontology, these findings point to doubts concerning the efficiency of manual tagging. Although there are some implementations that support the tagging with OntoClean meta-properties in existing ontology engineering environments, the number of actually tagged ontologies is obviously far too low. This again points to a discrepancy between the expected work and the expected benefit of using OntoClean. To turn OntoClean into a feasible and more often used ontology evaluation method, a far more precise and yet broader understandable description of OntoClean must become available, or else an approach for the automatic tagging of concepts must lower the time to tag ontologies dramatically. The latter approach requires far less training to the individual ontology engineer and evaluator.

[5] `http://proton.semanticweb.org`

The upper level ontology DOLCE [6] was created with the principles of OntoClean in mind. WordNet on the other hand was not created with ontological categories in mind, but rather adhering to linguistic structures. Aligning those two should reveal numerous errors in WordNet, by OntoClean standards, due to the different nature of the two. In [G+03], where this task is described, the authors say that the alignment of DOLCE and WordNet yielded almost only constraint violations regarding rigidity and much less on all other meta-properties. Thus it was essential to get reliable results for rigidity, more than for the other meta-properties.

Another problem is that tagging an ontology implies further ontological decisions possibly unintended by the ontology creators. Subjective point of views going further than the ontology is already committed to can be introduced through the tagging. For example, regarding the concept *Dalai Lama* we could state this concept is not rigid: a person is chosen to become the *Dalai Lama*. Thus a question of belief becomes relevant: buddhist religion claims that one does not become the *Dalai Lama*, but rather that one is the *Dalai Lama* since birth - or not. It is not a role a person plays, but rather it is the identity moving from body to body through the centuries. Simply tagging an ontology therefore reduces its possible audience by further ontological commitments.

We see that this contradicts to the definition of Rigidity, as there seem to be possible worlds where the concept is rigid and possible worlds in which it is not. Our approach dodges this problem by basing the taggings on statistics over a large corpus instead of an individual or small group's subjective point of view.

3.4 Results

One of the main findings of our experiments was that linguistic filtering really helps in the task of pattern-based ontology evaluation. As shown by tables 3.4, 3.5 and 3.7 without linguistic filtering the baseline for macro-average F-Measure was missed several times. And especially for *Identity* we noticed that the results could be improved by around 30% with the help of linguistic filtering. Another interesting result of the evaluation was that on average our system performed significantly better on the agreement, i.e. the intersection of two or three data sets, than on the single data sets. This is probably due to the fact that those concepts which were tagged identically by at least two of the human annotators are easier to tag – maybe, because they are less ambiguous.

The overall conclusion we draw from the evaluation of AEON was that despite the weaknesses of our pattern-based approach (see section 3.2) the first results are already very promising. Given the small amount of training data we had and the fact that we used standard Weka classifiers without much parameter tuning we hope to get even better results in future experiments.

[6] `http://www.loa-cnr.it/DOLCE.html`

	P		R		F				Classifier
	+	-	+	-	+	-	M-avg	**baseline**	no LF	
A_1	59.0	51.4	69.5	40.0	63.8	45.0	54.4	49.9	61.6	RandomForest
A_2	86.9	31.8	91.0	23.3	88.9	26.9	57.9	43.4	47.8	ADTree
A_3	76.5	23.5	76.1	24.0	76.3	23.8	50.1	**46.4**	**44.8**	RandomTree
avg	**74.1**	**35.6**	**78.9**	**29.1**	**76.3**	**31.9**	**54.1**	**46.6**	**51.4**	
A_1 / A_2	91.3	64.3	94.9	50.0	93.1	56.3	74.7	43.4	69.4	ADTree
A_1 / A_3	78.2	66.7	98.0	12.9	87.0	21.6	54.3	46.7	62.6	DecisionStump
A_2 / A_3	93.8	11.1	93.8	11.1	93.8	11.1	52.5	39.3	48.2	RandomTree
avg	**87.8**	**47.4**	**95.6**	**24.7**	**91.3**	**29.7**	**60.5**	**43.1**	**60.1**	
A_1 / A_2 / A_3	95.5	0.0	100.0	0.0	97.7	0.0	48.9	38.7	48.4	NBTree

Table 3.4: Rigidity (Best Results with Linguistic Filtering)

	P		R		F				Classifier
	+	-	+	-	+	-	M-avg	**baseline**	no LF	
A_1	75.0	34.8	84.1	23.2	79.3	27.8	53.6	47.2	49.5	ADTree
A_2	79.4	37.5	86.3	26.8	82.7	31.3	57.0	**46.2**	**45.0**	ADTree
A_3	87.3	55.0	95.8	26.8	91.4	36.1	63.8	43.4	47.9	RandomForest
avg	**80.6**	**42.4**	**88.7**	**25.6**	**84.5**	**31.7**	**58.1**	**45.6**	**47.5**	
A_1 / A_2	87.0	13.3	90.7	10.0	88.8	11.1	50.0	42.0	54.1	RandomTree
A_1 / A_3	93.0	46.7	95.2	36.8	94.1	41.2	67.7	40.5	50.8	NBTree
A_2 / A_3	95.7	57.1	98.1	36.4	96.9	44.4	70.7	38.6	48.2	ADTree
avg	**91.9**	**39.0**	**94.7**	**27.7**	**93.3**	**32.2**	**62.8**	**40.4**	**51.0**	
A_1 / A_2 / A_3	95.3	66.7	99.2	25.0	97.2	36.4	66.8	38.0	48.5	RandomForest

Table 3.5: Identity (Best Results with Linguistic Filtering)

	P		R		F				Classifier
	+	-	+	-	+	-	M-avg	**baseline**	no LF	
A_1	69.5	49.5	63.6	56.2	66.4	52.6	59.5	49.6	58.8	DecisionStump
A_2	43.0	61.2	46.0	58.3	44.4	59.7	52.1	47.7	57.8	ADTree
A_3	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
avg	**56.3**	**55.4**	**54.8**	**57.3**	**55.4**	**56.2**	**55.8**	**48.7**	**58.3**	
A_1 / A_2	57.6	53.6	51.5	59.7	54.4	56.5	55.5	50.0	60.2	ADTree
A_1 / A_3	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
A_2 / A_3	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
avg	**57.6**	**53.6**	**51.5**	**59.7**	**54.4**	**56.5**	**55.5**	**50.0**	**60.2**	
A_1 / A_2 / A_3	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 3.6: Unity (Best Results with Linguistic Filtering)

	P		R		F				Classifier
	+	-	+	-	+	-	M-avg	**baseline**	no LF	
A_1	68.2	40.9	69.8	39.1	69.0	40.0	54.5	**49.7**	**39.1**	RandomTree
A_2	30.0	81.5	23.1	86.3	26.1	83.8	55.0	41.8	56.7	RandomForest
A_3	100.0	0.0	100.0	0.0	100.0	0.0	50.0	34.7	50.0	ADTree
avg	**66.1**	**40.8**	**64.3**	**41.8**	**65.0**	**41.3**	**53.2**	**42.1**	**48.6**	
A_1 / A_2	45.5	70.0	45.5	70.0	45.5	70.0	57.8	**44.7**	**35.3**	ADTree
A_1 / A_3	100.0	0.0	100.0	0.0	100.0	0.0	50.0	35.1	40.0	ADTree
A_2 / A_3	100.0	0.0	100.0	0.0	100.0	0.0	50.0	N/A	50.0	ADTree
avg	**81.8**	**23.3**	**81.8**	**23.3**	**81.8**	**23.3**	**52.6**	**39.9**	**41.8**	
A_1 / A_2 / A_3	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 3.7: Dependence (Best Results with Linguistic Filtering)

Chapter 4

Conclusion and Future Work

We presented and evaluated two approaches for discovering changes in ontologies based on domain data analysis and formal ontology evaluation.

The first approach relies on the ontology learning framework Text2Onto in order to extract different types of ontology elements from a changing corpus. Since all ontology learning tasks are executed in an incremental manner, potential modifications to the ontology will be proposed whenever the corpus changes. A detailed evaluation of the ontology learning results showed very promising results, but also revealed a number of problems related to the evaluation procedure and individual user perception. In particular, we found a high disagreement among the human annotators especially with respect to the identification and the classification of instances. Apparently, an appropriate judgment of the learned ontology elements is at least as difficult as the task of ontology learning itself.

The second approach towards data-driven change discovery applies the OntoClean methodology in order to detect formal errors in a given taxonomy. For this purpose we developed AEON, the first and only framework for the automatic formal evaluation of ontologies. The evaluation of AEON yielded very good results, which could even be improved by the use of a larger training set and some parameter tuning of the classifiers.

Text2Onto and AEON can be seen as two complementary approaches being implemented independently of each other. Both rely on a linguistic analysis of natural language text in order to suggest possible changes in ontologies. Whereas Text2Onto has been developed to discover any kind of changes (including the addition of new concepts, instances or relations), the main purpose of AEON is to detect formal errors in the taxonomic hierarchy and to identify subclass-of relationships which have to be removed or modified in order to restore the formal correctness of the ontology.

For the future we are aiming at a detailed technical evaluation within the case studies with particular focus on the incremental ontology learning techniques. We will develop and evaluate a number of change strategies in order to customize the change discovery process with respect to different users and application scenarios. Moreover, we are planning to integrate both of our approaches with methods for usage-driven change discovery

within a common ontology evolution framework.

Appendix A

Text2Onto

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 3.0+
- WordNet version 2.0

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 3.0+[1] to <`GATE-DIR`>.
4. Install WordNet 2.0[2] to <`WN-DIR`>.

[1] `http://gate.ac.uk`
[2] `http://wordnet.princeton.edu/`

5. 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.

6. Make sure that you have installed the latest version of the Java virtual machine[3], since Text2Onto does not work with Java versions prior to 1.5.

7. 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 and WordNet.

8. 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.

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.

In order to start the ontology learning process, the user can select *File*$\rightarrow$ *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).

[3] `http://java.sun.com`

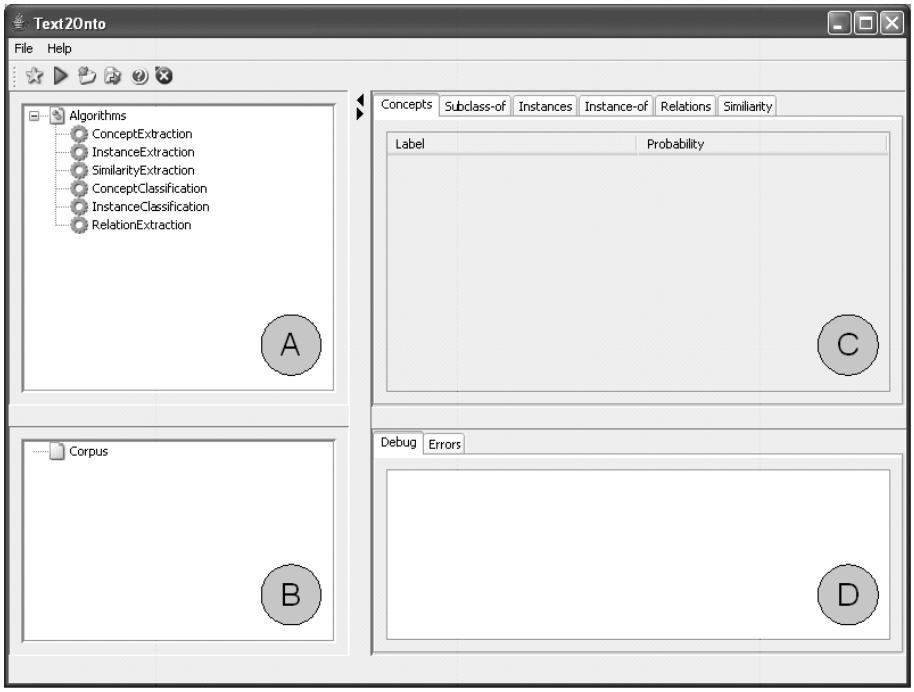

Figure A.1: GUI

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).

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 a simple KAON 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 KAONWriter( pom );
writer.write( new URI( "pom.kaon" ) );
```

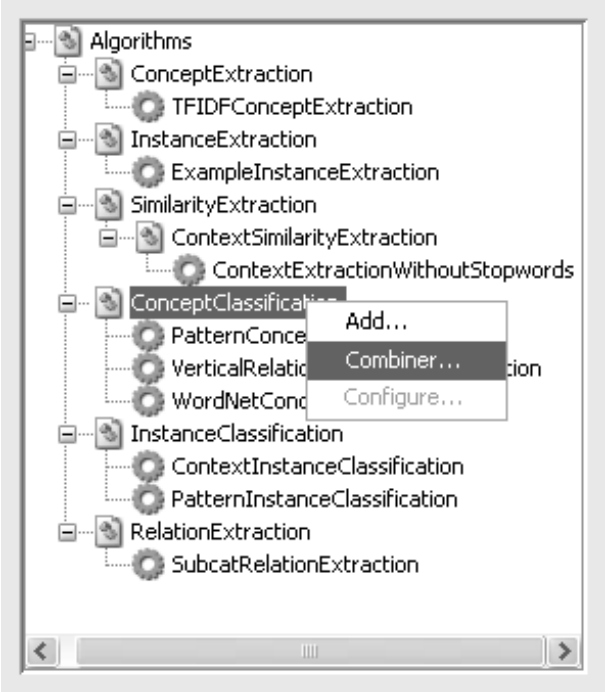

Figure A.2: Controller View

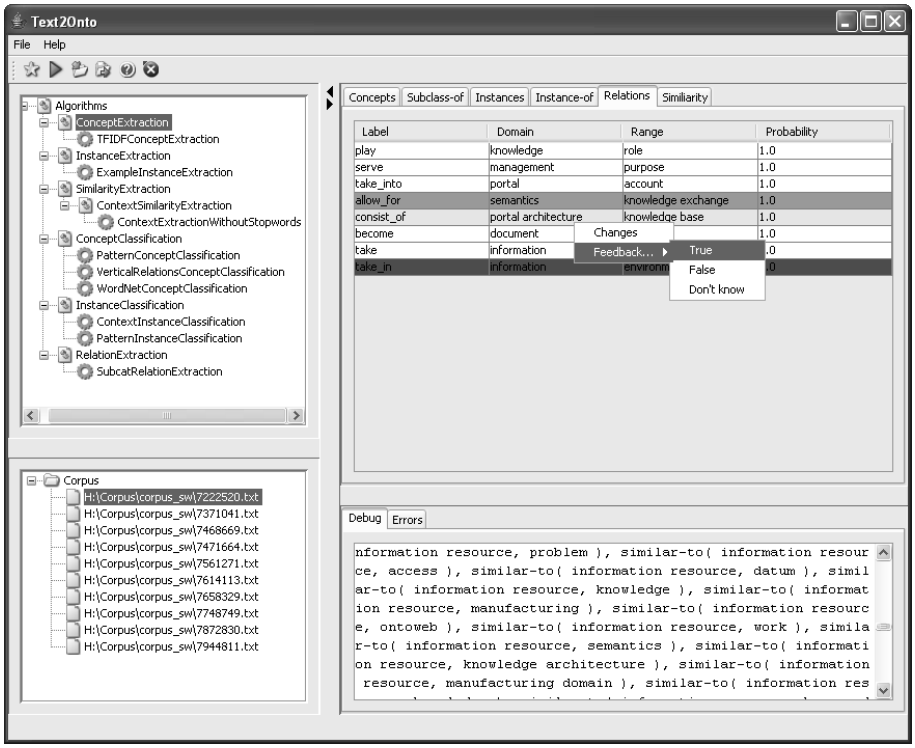

Figure A.3: User Feedback

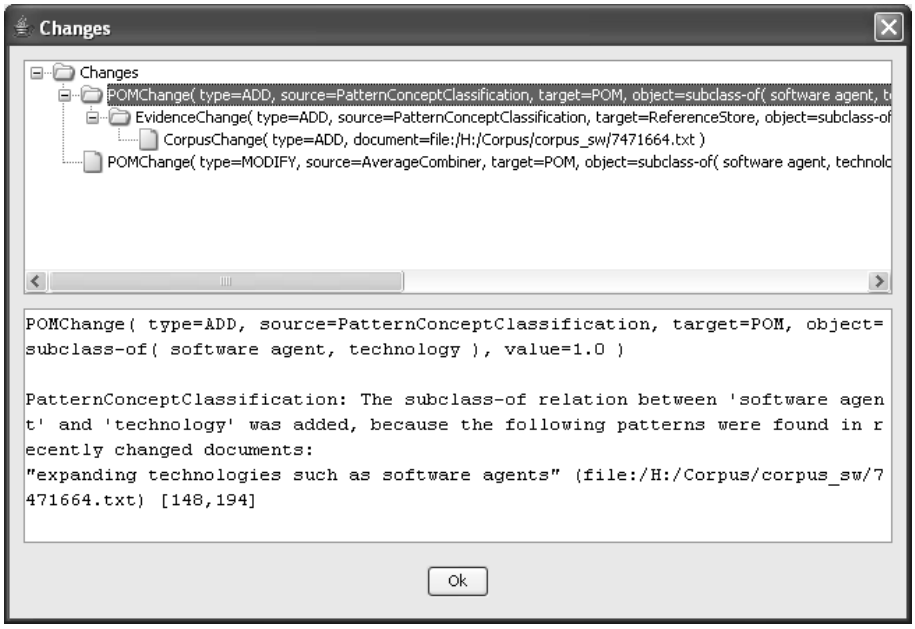

Figure A.4: Changes

Bibliography

[CB99] E. Charniak and M. Berland. Finding parts in very large corpora. In *Proceedings of the 37th Annual Meeting of the ACL*, pages 57–64, 1999.

[CHS04] P. Cimiano, S. Handschuh, and S. Staab. Towards the self-annotating web. In *Proceedings of the 13th World Wide Web Conference*, pages 462–471, 2004.

[CLS05] P. Cimiano, G. Ladwig, and S. Staab. Gimme' the context: Context-driven automatic semantic annotation with c-pankow. In *Proc. 14th WWW*. ACM, 2005.

[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.

[CPSTS05] P. Cimiano, A. Pivk, L. Schmidt-Thieme, and S. Staab. Learning taxonomic relations from heterogeneous sources of evidence. In *Ontology Learning from Text: Methods, Applications and Evaluation*. IOS Press, 2005.

[CV05] 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.

[G+03] A. Gangemi et al. Sweetening WordNet with Dolce. *AI Magazine*, Fall 2003.

[Gra90] R. M. Gray. *Entropy and Information Theory*. Springer, 1990.

[Gre99] G. Grefenstette. The WWW as a resource for example-based MT tasks. In *Proceedings of ASLIB'99 Translating and the Computer 21*, 1999.

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

[GW04] N. Guarino and C. A. Welty. An overview of OntoClean. International Handbooks on Information Systems, chapter 8, pages 151–172. Springer, 2004.

[Hea92] M.A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics*, pages 539–545, 1992.

[HS98] U. Hahn and K. Schnattinger. Ontology engineering via text understanding. In *Proceedings of the 15th World Computer Congress 'The Global Information Society on the Way to the Next Millenium' (IFIP'98)*, 1998.

[HS05] P. Haase and Y. Sure. Ontology mangement and evolution - evaluation. SEKT deliverable 3.1.2, Institute AIFB, University of Karlsruhe, 2005.

[KF98] J. Tsuji K. Frantzi, S. Ananiadou. The c-value/nc-value method of automatic recognition for multi -word terms. In *Proceedings of the ECDL*, pages 585–604, 1998.

[KLO02] F. Keller, M. Lapata, and O. Ourioupina. Using the web to overcome data sparseness. In *Proceedings of EMNLP-02*, pages 230–237, 2002.

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

[MS00] Alexander Maedche and Steffen Staab. Discovering conceptual relations from text. In W. Horn, editor, *Proceedings of the 14th European Conference on Artificial Intellignece (ECAI'2000)*, 2000.

[RS03] P. Resnik and N. Smith. The web as a parallel corpus. *Computational Lingusitics*, 29(3):349–380, 2003.

[Sal91] G. Salton. Developments in automatic text retrieval. 253:974–979, 1991.

[SAS03] Y. Sure, J. Angele, and S. Staab. OntoEdit: Multifaceted inferencing for ontology engineering. *Journal on Data Semantics*, LNCS(2800):128–152, 2003.

[SC99] Mark Sanderson and W. Bruce Croft. Deriving concept hierarchies from text. In *Research and Development in Information Retrieval*, pages 206–213, 1999.

[SWM04] M. K. Smith, C. Welty, and D. McGuinness. OWL Web Ontology Language Guide, 2004. W3C Recommendation 10 February 2004, available at http://www.w3.org/TR/owl-guide/.

[VNCN05] P. Velardi, R. Navigli, A. Cuchiarelli, and F. Neri. Evaluation of ontolearn, a methodology for automatic population of domain ontologies. In P. Buitelaar, P. Cimiano, and B. Magnini, editors, *Ontology Learning from Text: Methods, Applications and Evaluation*. IOS Press, 2005. to appear.

[VS05] J. Voelker and Y. Sure. Data-driven change discovery. SEKT deliverable 3.3.1, Institute AIFB, University of Karlsruhe, 2005.

[VVS05] Johanna Völker, Denny Vrandecic, and York Sure. Automatic evaluation of ontologies (aeon). In Y. Gil, E. Motta, V. R. Benjamins, and M. A. Musen, editors, *Proceedings of the 4th International Semantic Web Conference (ISWC2005)*, volume 3729 of *LNCS*, pages 716–731. Springer Verlag Berlin-Heidelberg, NOV 2005.

[WMCC04] C. Welty, R. Mahindru, and J. Chu-Carroll. Evaluating ontology cleaning. In D.L. McGuinness and G. Ferguson, editors, *AAAI2004*. AAAI / MIT Press, 2004.