D3.3.1 Data-driven Change Discovery

Johanna Völker and York Sure
(Institute AIFB, University of Karlsruhe)

Abstract.
EU-IST Integrated Project (IP) IST-2003-506826 SEKT
Deliverable D3.3.1 (WP3.3)
In this deliverable we present work performed in the task ‘T3.3 Data-driven Change Discovery’. Text2Onto is a framework for data-driven change discovery by incremental ontology learning. It uses natural language processing and text mining techniques in order to extract an ontology from text and provides support for the adaptation of the ontology over time as documents are added or removed. Explicit modeling of all kinds of changes and an explanation component guarantee maximum transparency and traceability of the ontology learning process.
Keyword list: Data-driven Change Discovery, Ontology Learning, Ontology Management

Document Id.	SEKT/2005/D3.3.1/v1.0
Project	SEKT EU-IST-2003-506826
Date	July 22, 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

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

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 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 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

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

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

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

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

Sirma AI EAD, 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

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

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 work performed in the task 'T3.3 Data-driven Change Discovery'. We present Text2Onto which is a framework for data-driven change discovery by incremental ontology learning. It uses natural language processing and text mining techniques in order to extract an ontology from text and provides support for the adaptation of the ontology over time as documents are added or removed. Explicit modeling of all kinds of changes and an explanation component guarantee maximum transparency and traceability of the ontology learning process.

The deliverable targets different audiences by covering topics such as user guidance for end users (e.g. the SEKT case study partners) and conceptual issues and architecture for developers (e.g. the SEKT technical partners).

Contents

Chapter 1

Introduction

1.1 Motivation

Providing a shared conceptualization of a domain of interest ontologies have become a key technology for data interchange and integration. They have shown a great potential for modeling knowledge which can be used by various types of applications in areas such as information retrieval and extraction, web services, natural language processing, clustering and categorization.

But the acquisition of knowledge about a particular domain remains a bottleneck in the process of developing knowledge-based applications. Since especially very complex domains are often extensively described by collections of text documents, many tools have been developed to support the ontology engineering process by text mining techniques. Nevertheless, these tools typically ignore the highly dynamic character of the data. In case of changes to the underlying data the ontology becomes outdated and after a while a complete re-engineering of the ontology might be unavoidable. Continuous and expensive modeling efforts by ontology engineers are required to preserve both domain coverage and consistency.

Text2Onto is a framework for data-driven change discovery by incremental ontology learning. It uses natural language processing and text mining techniques in order to extract an ontology from text and provides support for the adaptation of the ontology over time as documents are added or removed. Explicit modeling of all kinds of changes and an explanation component guarantee maximum transparency and traceability of the ontology learning process.

1.2 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.

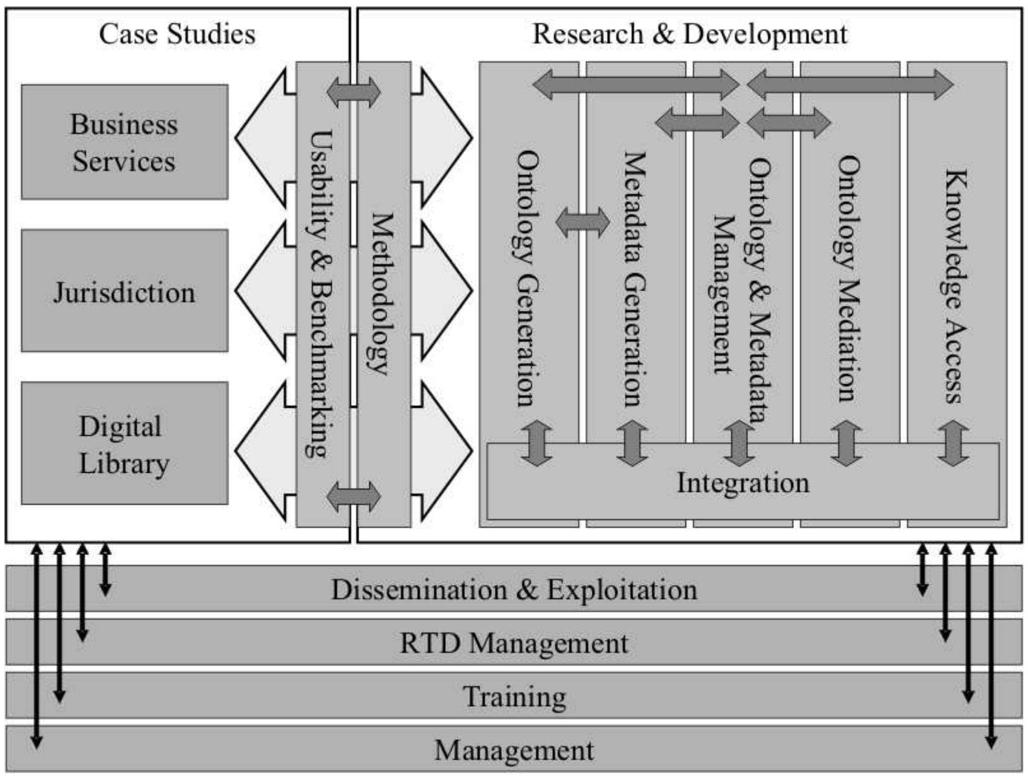


Figure 1.1: The SEKT Big Picture

1.3 Relevance for SEKT case studies

WP10 - Legal Case Study. The goal of the case study is to provide support to professional judges. Therefore, Iuriservice has been developed - an improved FAQ searching system design based on Spanish NLP, background calculation of ontologies, caching of background calculated data, and a multistage searching approach with progressive delimitation of the FAQ target. A user (a judge in her first appointment) types a question and expects the system to find out an FAQ candidate as close as possible to the question.

The system uses several legal domain ontologies to obtain additional understanding of the user's question. The system tries to find out a track of the user's question or part of it in the ontology. Since in the past, these ontologies were actually built by human experts who capture the knowledge of the legal domain, there is a great potential for improving the efficiency of the knowledge acquisition process by ontology learning techniques.

WP11 - BT Case Study. The main purpose of this case study is to investigate the use of Semantic Web technologies for searching, browsing and maintaining BT's Digital Library. Built in 1994 and continuously developed up to the present, the library offers its users a single interface to different databases containing both PDF and HTML documents from a variety of publishers. 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 personalized alerts and so-called information spaces are among the main features of the Digital Library. Each information space consists of a number of documents related to one or more topics.

In order to make use of Semantic Web technology at least two kinds of ontologies are intended to be integrated into the Digital Library: On the one hand, a global domain ontology will be used to categorize the information spaces, i.e. each information space will be associated with one or more concepts within the topic hierarchy. On the other hand, a more specialized domain ontology might be extracted from each information space giving the user a much more detailed impression of its content. Moreover, provided that concepts, instances and relations within such an individual ontology are linked to paragraphs, sentences or words in different information space documents, the ontology might significantly improve browsing and searching the information space.

1.4 Organization

The deliverable targets different audiences by covering topics such as user guidance for end users (e.g. the SEKT case study partners) and conceptual issues and architecture for developers (e.g. the SEKT technical partners).

More specifically, we begin in the next chapter 2 with a user guide showing how to install and use Text2Onto. The next chapter 3 provides the conceptual baseline for data-driven change discovery and enumerates requirements and implementation issues for Text2Onto which stem from the conceptual idea. Chapter 4 focuses on the architecture including topics such as the underlying ontology model and the usage of natural language processing in Text2Onto. Before concluding and presenting future work in chapter 6 we discuss related work in chapter 5.

Chapter 2

User Guide

In this chapter we provide detailed information on how to install Text2Onto and how to use the graphical user interface as well as the API.

2.1 Installation (Windows)

1. Download Text2Onto from `http://ontoware.org/projects/text2onto/`.
2. Unzip file to <`T2O-DIR`> (e.g. `c:\text2onto`).
3. Install GATE[1] to <`GATE-DIR`>.
4. Install WordNet [2] to <`WN-DIR`>.
5. Modify <`T2O-DIR`>`/text2onto.properties`[3].

Listing 2.1: text2onto.properties

```
language=english
gate_dir=<T2O-DIR>/3rdparty/gate/
gate_app=application.gate
jape_main=main.jape
stop_file=stopwords.txt
creole_dir=<T2O-DIR>/3rdparty/gate/
jwnl_properties=<T2O-DIR>/3rdparty/jwnl/file_properties
    .xml
```

[1] `http://gate.ac.uk`

[2] `http://wordnet.princeton.edu/`

[3] Please note, that in listing 2.1 the variable `gate_dir` does not point to the installation directory of GATE, but to a directory containing Text2Onto-specific GATE resources such as JAPE patterns and applications for different languages.

```
temp_corpus=c:/temp/text2onto/
algorithms=<T2O-DIR>/algorithms.xml
icons=<T2O-DIR>/icons/
```

6. Modify <`T2O-DIR`>`/3rdparty/jwnl/file_properties.xml`.

Listing 2.2: file_properties.xml

```
[...]
<param name="file_manager" value="net.didion.jwnl.
   dictionary.file_manager.FileManagerImpl">
      <param name="file_type" value="net.didion.jwnl.
         princeton.file.
         PrincetonRandomAccessDictionaryFile"/>
      <param name="dictionary_path" value="<WN-DIR>\
         dict"/>
</param> [...]
```

7. Modify <`T2O-DIR`>`/text2onto.bat`.

Listing 2.3: text2onto.bat

```
@echo off
set T2O=<T2O-DIR>
set LIB=<T2O-DIR>\3rdparty
set GATE=<GATE-DIR>
[...]
```

8. Please, make sure that you have installed the latest version of the Java virtual machine[4], since Text2Onto does not work with Java versions prior to 1.5.

9. Start `text2onto.bat`.

2.2 Usage Scenario

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

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

corpus gets preprocessed by the natural language processing component described in section 4.1.2, 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* (see section 4.1.1) to its caller, which could be the algorithm controller, but also a more complex algorithm (cf. section 4). After the process of ontology extraction is finished, the POM is presented to the user.

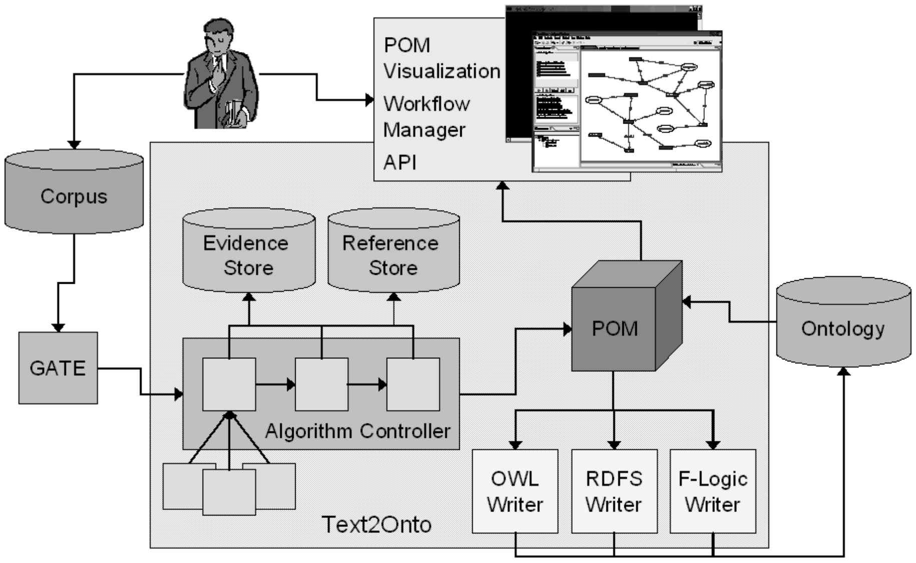


Figure 2.1: Scenario

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.

2.3 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

2.2).

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 2.3).

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.

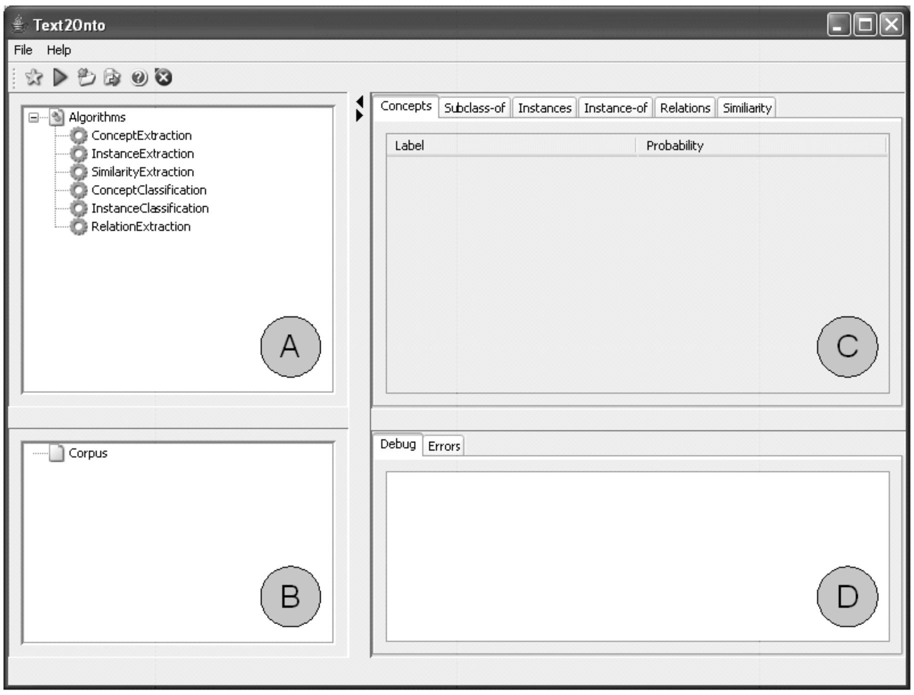


Figure 2.2: GUI

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 prim-

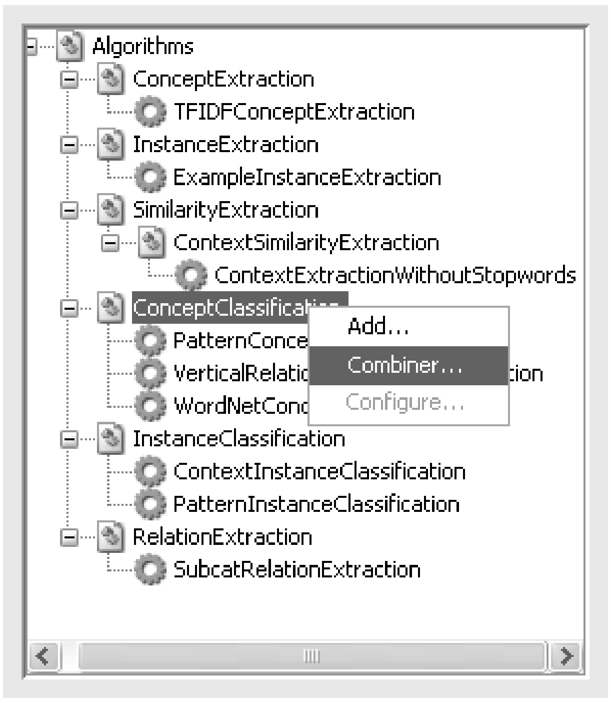


Figure 2.3: Controller View

itives are displayed to the user, who can interact with the POM by giving feedback to individual learning results (cf. figure 2.4).

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 2.5).

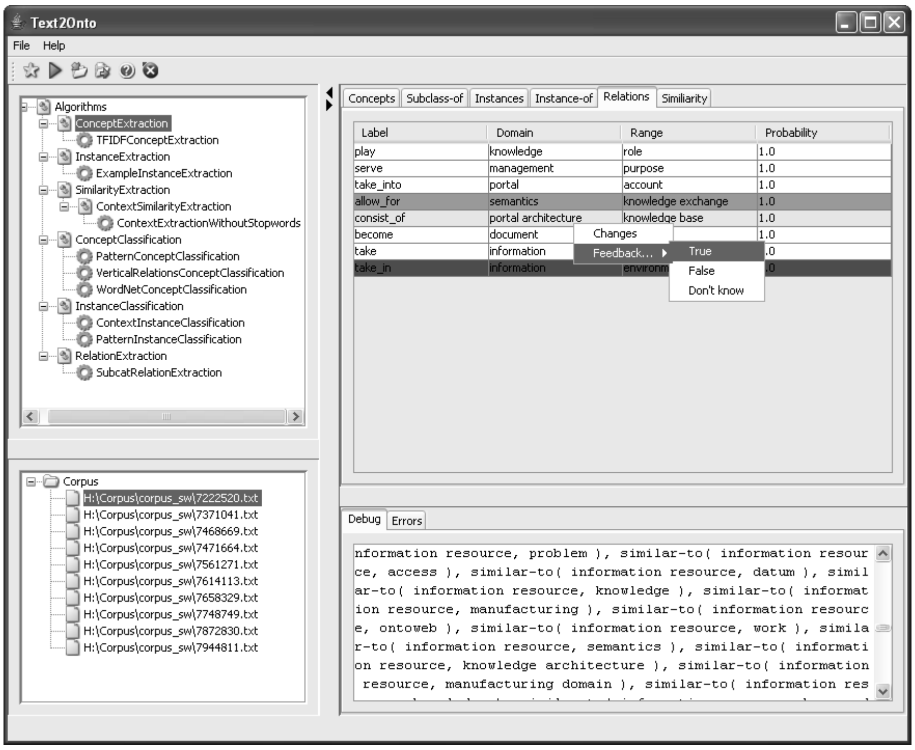
2.4 API

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 2.4) 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 2.4: 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" ) );
```



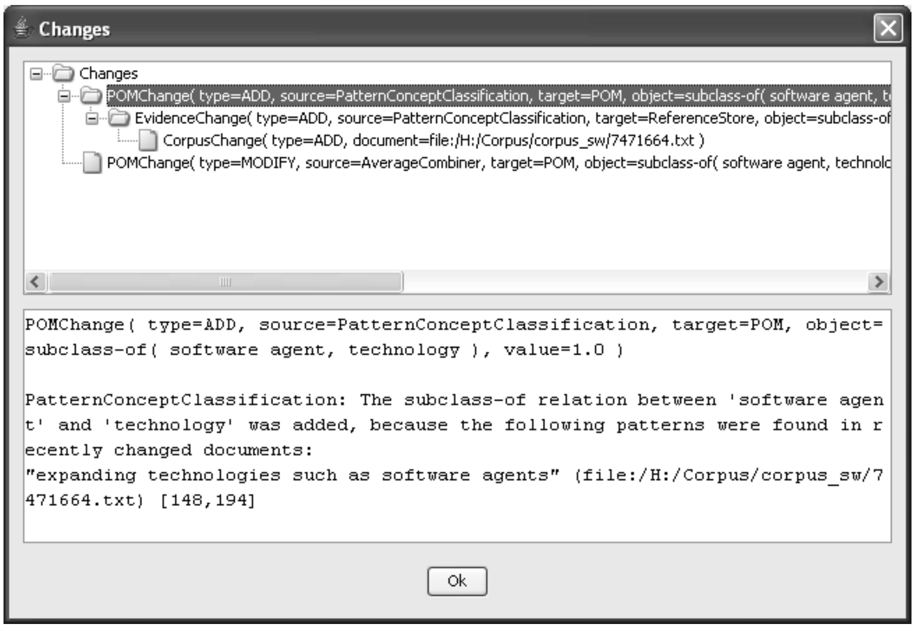
```
Algorithms
  ConceptExtraction
    TFIDFConceptExtraction
  InstanceExtraction
    ExampleInstanceExtraction
  SimilarityExtraction
    ContextSimilarityExtraction
      ContextExtractionWithoutStopwords
  ConceptClassification
    PatternConceptClassification
    VerticalRelationsConceptClassification
    WordNetConceptClassification
  InstanceClassification
    ContextInstanceClassification
    PatternInstanceClassification
  RelationExtraction
    SubcatRelationExtraction
```

```
Corpus
  H:\Corpus\corpus_sw\7222520.txt
  H:\Corpus\corpus_sw\7371041.txt
  H:\Corpus\corpus_sw\7468669.txt
  H:\Corpus\corpus_sw\7471664.txt
  H:\Corpus\corpus_sw\7561271.txt
  H:\Corpus\corpus_sw\7614113.txt
  H:\Corpus\corpus_sw\7658329.txt
  H:\Corpus\corpus_sw\7748749.txt
  H:\Corpus\corpus_sw\7872830.txt
  H:\Corpus\corpus_sw\7944811.txt
```

Label	Domain	Range	Probability
play	knowledge	role	1.0
serve	management	purpose	1.0
take_into	portal	account	1.0
allow_for	semantics	knowledge exchange	1.0
consist_of	portal architecture	knowledge base	1.0
become	document		1.0
take	information		.0
take_in	information	environm	.0

```
nformation resource, problem ), similar-to( information resour
ce, access ), similar-to( information resource, datum ), simil
ar-to( information resource, knowledge ), similar-to( informat
ion resource, manufacturing ), similar-to( information resourc
e, ontoweb ), similar-to( information resource, work ), simila
r-to( information resource, semantics ), similar-to( informati
on resource, knowledge architecture ), similar-to( information
 resource, manufacturing domain ), similar-to( information res
```

Figure 2.4: User Feedback



```
Changes
  POMChange( type=ADD, source=PatternConceptClassification, target=POM, object=subclass-of( software agent, t
    EvidenceChange( type=ADD, source=PatternConceptClassification, target=ReferenceStore, object=subclass-of
      CorpusChange( type=ADD, document=file:/H:/Corpus/corpus_sw/7471664.txt )
  POMChange( type=MODIFY, source=AverageCombiner, target=POM, object=subclass-of( software agent, technolo
```

```
POMChange( type=ADD, source=PatternConceptClassification, target=POM, object=
subclass-of( software agent, technology ), value=1.0 )

PatternConceptClassification: The subclass-of relation between 'software agen
t' and 'technology' was added, because the following patterns were found in r
ecently changed documents:
"expanding technologies such as software agents" (file:/H:/Corpus/corpus_sw/7
471664.txt) [148,194]
```

Figure 2.5: Changes

Chapter 3

Data-driven Change Discovery

In order to define the task of data-driven change discovery we first distinguish between *change capturing* and *change discovery*.

Change capturing can be defined as the generation of ontology changes from explicit and implicit requirements. Explicit requirements are generated, for example, by ontology engineers who want to adapt the ontology to new requirements or by the end-users who provide the explicit feedback about the usability of ontology entities. The changes resulting from this kind of requirements are called top-down changes. Implicit requirements leading to so-called bottom-up changes are reflected in the behavior of the system and can be induced by applying change discovery methods.

Change discovery aims at generating implicit requirements by inducing ontology changes from existing data. [Sto04] defines three types of change discovery: (i) structure-driven, (ii) usage-driven and (iii) data-driven. Whereas structure-driven changes can be deduced from the ontology structure itself, usage-driven changes result from the usage patterns created over a period of time. Data-driven changes are generated by modifications to the underlying data, such as text documents or a database, representing the knowledge modeled by an ontology. Therefore, *data-driven change discovery* provides methods for automatic or semi-automatic adaption of an ontology according to modifications being applied to the underlying data set.

The benefits of data-driven change discovery are twofold. First, an elaborated change management system enables the user to explicitly track the changes to the ontology since the last change in the document collection thus being able to trace the evolution of the ontology with respect to changes in the underlying document collection. Second and even more important, there is no longer the need for processing the whole document collection when it changes thus leading to increased efficiency.

3.1 Requirements

Independently from a particular application scenario some requirements have to be met by any application which is designed to support data-driven change discovery.

The most important one is, of course, the need to keep track of all changes to the data. Each change must be represented in a way which allows for associating with it various kinds of information, such as its type, the source it has been created from and its target object (e.g. a text document). In order to make the whole system as transparent as possible not only changes to the data set, but also changes to the ontology should be logged. If ontological changes are caused by changes to the underlying data, the former should be associated with information about the corresponding modification to the data. Moreover, the system should allow for the definition of various *change strategies* which specify the degree of influence of the data changes with respect to the ontology. This permits to take into account the confidence the user has in different data sources or the fact that documents might become out-dated after a while. For example, a user might want the ontology to be updated in case of newly added or modified data, but, on the other hand, he might want the ontology to remain unchanged if some part of the data set is deleted.

It is quite obvious that automatic or semi-automatic data-driven change discovery requires a formal, explicit representation of two kinds of knowledge: First, knowledge about which concepts, instances and relations are affected by certain changes to the data and second, knowledge about how to react to these changes in an appropriate way, i.e. how to update the ontology in response to these changes. Since most of this knowledge is usually unavailable in case of a manually built ontology, we can conclude that an implementation of data-driven change discovery methods should be embedded in the context of an ontology extraction system. Such systems usually represent general knowledge about the relationship between an ontology and the underlying data set by means of ontology learning algorithms. Consequently, the concrete knowledge to be stored by an ontology extraction system depends on the way these algorithms are implemented. A concept extraction algorithm, for example, might need to store the text references and term frequencies associated with each concept, whereas a pattern-based concept classification algorithm might have to remember the occurrences of all hyponymy patterns matched in the text.

Whereas existing tools such as TextToOnto mostly neglect this kind of concrete knowledge and therefore do not provide any support for data-driven change discovery, the next generation of ontology extraction systems will explicitly target the problem of incremental ontology learning.

3.2 Implementation

Text2Onto meets the requirements described by providing a flexible change management architecture. The main characteristics of this architecture are (i) explicit modeling of changes to the corpus, the evidences, the reference repository and the POM and (ii) Incremental Ontology Learning. Figure 3.1 provides an abstract overview of the current implemenation of the change management framework.

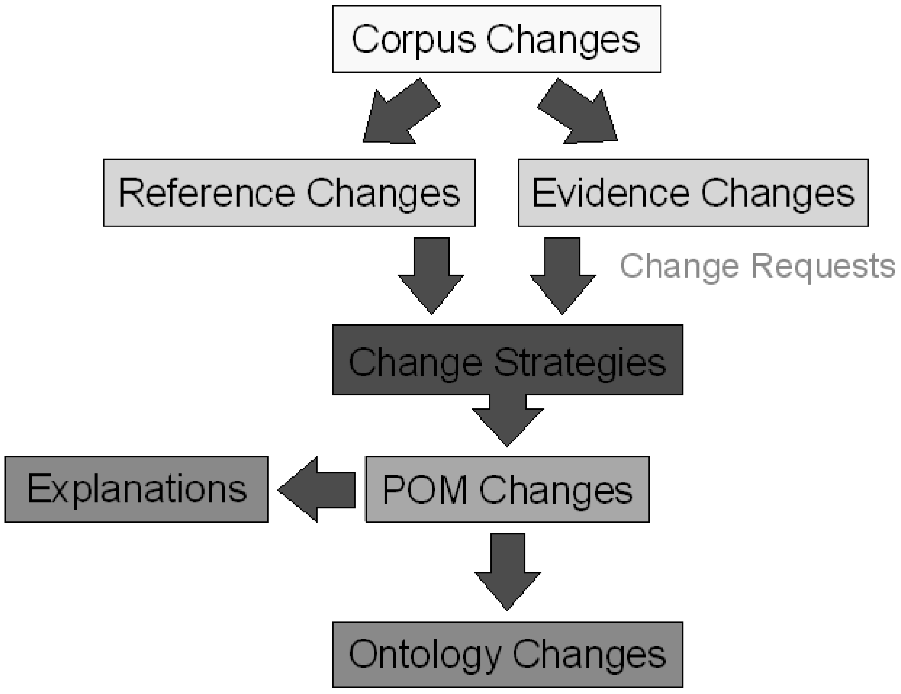


Figure 3.1: Change Management

In case of changes to the corpus, i.e. the addition or removal of a document, each algorithm updates its evidences for the existence of all affected ontology elements. Moreover, references are generated or removed linking each ontology element to its lexicalizations in the corpus. These references not only facilitate the generation of explanations for ontology changes, but they can also be used for creating semantic annotations and, most importantly, they allow for quickly determine which ontology elements are affected by certain changes to the corpus.

Once all evidences and references have been updated the algorithm creates suggestions for changes to the POM which may be processed by user-defined change strategies before finally being applied to the POM. These change strategies can be used, for example, to model the extend to which particular types of ontology elements are affected by changes to the corpus or the influence of the changing up-to-dateness of documents on the ontology.

Finally, based on recent changes to the POM a concrete ontology is created or modified by one of the available ontology writers for OWL, KAON or RDFS, which can be configured to take into account a set of user-specified consistency conditions.

Please note that maximum traceability is garanteed by the fact that explanations are generated for each change to the POM. All explanations are available in natural language as well as in machine-understandable form, which will allow for a tight integration with the DILIGENT methdology at a later development stage.

Chapter 4

Architecture

4.1 Overview

The architecture of Text2Onto (cf. figure 4.1) is centered around the Probabilistic Ontology Model (see Section 4.1.1) which stores the results of the different ontology learning algorithms (cf. section 4.2). The algorithms are initialized by a controller, the purpose of which is (i) to trigger the linguistic preprocessing of the data, (ii) to execute the ontology learning algorithms in the appropriate order and (iii) to apply the algorithms' change requests to the POM. The fact that none of the algorithms has permission to manipulate the POM directly guarantees maximum transparency and allows for the flexible composition of arbitrarily complex algorithms as described below.

The execution of each algorithm consists of three phases: First, in the *notification phase*, the algorithm learns about recent changes to the corpus. Second, in the *computation phase*, these changes are mapped to changes with respect to the reference repository, which stores all kinds of knowledge about the relationship between the ontology and the data (e.g. pointers to all occurrences of a concept). And finally, in the *result generation phase*, requests for POM changes are generated from the updated content of the reference repository.

The algorithms provided by the Text2Onto framework can be classified according to two different aspects: *task*, i.e. the kind of modeling primitives (see section 4.1.1) they produce, and *type*, that means the method which is employed in order to extract instances of the regarding primitives from the text. Each algorithm producing a certain kind of modeling primitive can be configured to apply several algorithms of different types and to combine their requests for POM changes in order to obtain a more reliable probability for each instantiated primitive (cf. [CPSTS04]). Various types of pre-defined strategies allow for specifying the way the individual probabilities are combined. An algorithm for the extraction of instance-of relationships, for example, might be configured to apply

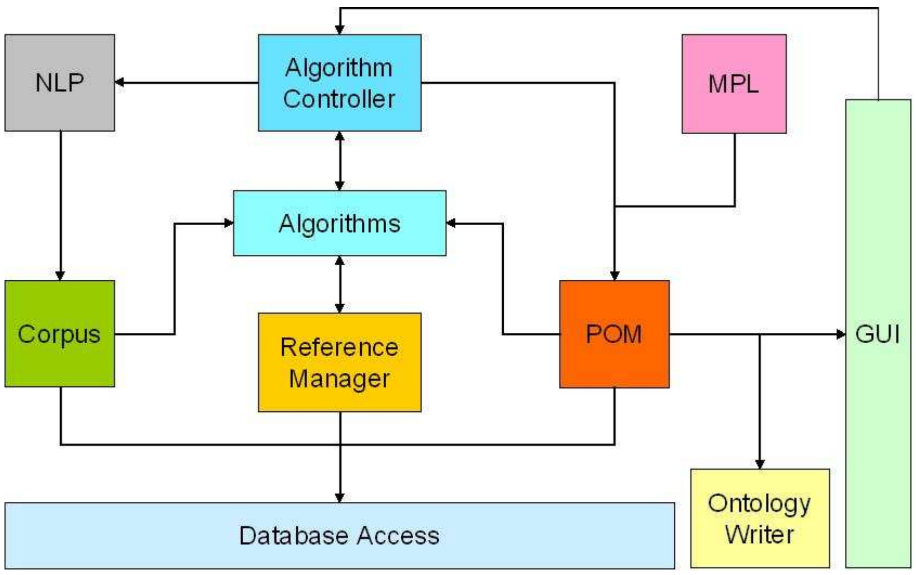


Figure 4.1: Architecture of Text2Onto

a context-based instance classification as well as an algorithm which queries Google[1] for manually defined patterns. The results of both algorithms can then be averaged, for instance, or combined in a way which prefers the most reliable among the algorithms.

4.1.1 The Probabilistic Ontology Model

A *Preliminary Ontology Model* (POM) as used by Text2Onto is a collection of instantiated modeling primitives which are independent of a concrete ontology representation language. In fact, Text2Onto includes a *Modeling Primitive Library* (MPL) which defines these primitives in a declarative fashion. The obvious benefits of defining primitives in such a declarative way are twofold. On the one hand, adding new primitives does not imply changing the underlying framework thus making it flexible and extensible. On the other hand, the instantiated primitives can be translated into any knowledge representation language, given that the expressivity of the primitives does not exceed the expressivity of this target language. Thus, the POMs learned by Text2Onto can be translated into various ontology representation languages such as RDFS[2], OWL [3] and F-Logic [KLW95]. In fact we follow a similar approach to knowledge representation as advocated in [Gru93] and [SEM01]. Gruber as well as Staab et al. adopt a translation approach to knowledge engineering in which knowledge is modeled at a meta-level rather than in a particular knowledge representation language and is then translated into different target languages. While Gruber's Frame Ontology is based on KIF [Gen91], Staab et al. ground their work on the RDFS model as it represents a core which is supported by most knowledge repre-

[1]http://www.google.com
[2]http://www.w3.org/TR/rdf-schema/
[3]http://www.w3.org/TR/owl-features/

sentation and logic languages. In Text2Onto we follow this translation-based approach to knowledge engineering and define the relevant modeling primitives in the MPL. So called *ontology writers* are then responsible for translating instantiated modeling primitives into a specific target knowledge representation language. The modeling primitives we use in Text2Onto are given below.

- concepts
- concept inheritance (`subclass-of`)
- concept instantiation (`instance-of`)
- general properties / relations (`related-to`)
- mereological relations (`part-of`)
- domain and range restrictions
- equivalence (`similar-to`)

It is important to mention that the above list is in no way exhaustive and could be extended whenever it is necessary. In particular, support for the extraction of additional relation types other than `part-of` (as a special case of the general `related-to` relationship) could be provided by future versions of Text2Onto.

Every instance of one of these modeling primitives gets assigned a number of rating annotations indicating how certain the algorithm in question is about the correctness or relevance of the corresponding instance. The purpose of these ’probabilities’ is to facilitate the user interaction by allowing her to filter the POM and thereby select only a number of relevant instances of modeling primitives to be translated into a target language of her choice.

4.1.2 Natural Language Processing

Many existing ontology learning environments focus either on pure machine learning techniques [BNC00] or rely on linguistic analysis [BOS03, VNCN05] in order to extract ontologies from natural language text. Text2Onto combines machine learning approaches with basic linguistic processing such as tokenization or lemmatizing and shallow parsing. Since it is based on the GATE framework [CMBT02] it is very flexible with respect to the set of linguistic algorithms used, i.e. the underlying GATE application can be freely configured by replacing existing algorithms or adding new ones such as a deep parser if required. Another benefit of using GATE is the seamless integration of JAPE [CMT00] which provides finite state transduction over annotations based on regular expressions.

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 and a stemmer respectively.

After the basic linguistic preprocessing is done, a JAPE transducer is run over the annotated corpus in order to match a set of particular patterns required by the ontology learning algorithms. Whereas the left hand side of each JAPE pattern defines a regular expression over existing annotations, the right hand side describes the new annotations to be created (see listing 4.2). For Text2Onto we developed JAPE patterns for both shallow parsing and the identification of modeling primitives, e.g. concepts, instances and different types of relations (c.f. [Hea92]).

Listing 4.1: JAPE pattern: Noun Phrase

```
Rule: NounPhrase (
 (
  {Token.category == DT}
  {SpaceToken.kind == space}
 )?
 (
  ({Token.category == JJ}|{Token.category == VBG})
  {SpaceToken.kind == space}
 )*
 (
  (
   ({Token.category == NN} | {Token.category == NNS})
   {SpaceToken.kind == space}
  )*
  ({Token.category == NN} | {Token.category == NNS})
 ):np_head
):np
-->
:np.NounPhrase = { rule = "NounPhrase" },
:np_head.Head = { rule = "NounPhrase" }
```

Since obviously, both types of patterns are language specific, different sets of patterns for shallow parsing and ontology extraction have to be defined for each language. Because of this and due to the fact that particular processing components for GATE have to be available for the regarding language, Text2Onto currently only supports ontology learning from English texts. Fortunately, thanks to recent research efforts made in the SEKT project[4] GATE components for the linguistic analysis of various languages such as German and Spanish [May05] have been made available recently. Since we want to

[4]www.sekt-project.com

Listing 4.2: JAPE pattern: Hearst

```
Rule: Hearst_1 (
 (NounPhrase1):superconcept
 (
  {Token.kind == punctuation}
 )?
 {SpaceToken.kind == space}
 {Token.string == "such"}
 {SpaceToken.kind == space}
 {Token.string == "as"}
 {SpaceToken.kind == space}
 (NounPhrasesAlternatives):subconcept
):hearst1
-->
:hearst1.SubclassOfRelation = { rule = "Hearst1" },
:subconcept.Domain = { rule = "Hearst1" },
:superconcept.Range = {rule = "Hearst1" }
```

provide full support for all of these languages in future releases of Text2Onto, we have already integrated some of these components, and we are currently working on the development of appropriate patterns for Spanish and German.

4.2 Algorithms

This section briefly describes for each modeling primitive the algorithms used to learn corresponding instances thereof. In particular we describe the way the probability for an instantiated modeling primitive is calculated.

4.2.1 Concepts

In Text2Onto we have implemented several measures to assess the relevance of a certain term with respect to the corpus in question. In particular, we implemented different algorithms calculating the following measures: Relative Term Frequency (RTF), TFIDF (Term Frequency Inverted Document Frequency) [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 probability in the POM.

4.2.2 Subclass-of Relations

In order to learn subclass-of relations, in Text2Onto we have implemented various algorithms using different kinds of sources and techniques following the approach in [CPSTS04]. In particular we implemented algorithms exploiting the hypernym structure of WordNet [Mil95], matching Hearst patterns [Hea92] in the corpus as well as in the World Wide Web and applying linguistic heuristics mentioned in [VNCN05]. The results of the different algorithms are then combined through combination strategies as described in [CPSTS04]. This approach has been evaluated with respect to a collection of tourism-related texts by comparing the results with a reference taxonomy for this domain. The best result obtained was an F-Measure of 21.81%, a precision of 17.38% and a recall of 29.95%. As the algorithm already indicates the confidence in its prediction with a value between 0 and 1, the probability given in the POM can be set accordingly.

4.2.3 Mereological Relations

For the purpose of discovering mereological (part-of) relations in the corpus, we developed JAPE expressions matching the patterns 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 probability is then calculated by dividing by the sum of occurrences of patterns in which t_1 appears as a part. Further, as in the algorithm described above we also consult WordNet for mereological relations and combine the elementary probabilities with a certain combination strategy.

4.2.4 General Relations

In order to learn general relations, Text2Onto employs a shallow parsing strategy to extract subcategorization frames enriched with information about the frequency of the terms appearing as arguments. In particular, it extracts the following syntactic frames:

- transitive, e.g. love(subj,obj)
- intransitive + PP-complement, e.g. walk(subj,pp(to))
- transitive + PP-complement, e.g. hit(subj,obj,pp(with))

and maps this subcategorization frames to ontological relations. For example, given the following enriched subcategorization frame

hit(subj:*person*,obj:*thing*,with:*object*)

the system would update the POM with these relations:

hit(domain:*person*,range:*thing*)
hit_with(domain:*person*,range:*object*)

The probability of the relation is then 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 in question.

4.2.5 Instance-of Relations

In order to assign instances or named entities appearing in the corpus to their correct 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 as in [AM02]. As similarity measure we use the Skewed divergence presented in [Lee99] as it was found to perform best in our experiments. Using this similarity measure as well as further heuristics, we achieved an F-Measure of 32.6% when classifying instances with respect to an ontology comprising 682 concepts [CV04]. Alternatively, we also implemented a pattern-matching algorithm similar to the one used for discovering part-of relations (see above).

Equivalence

Following the assumption that terms or concepts are equivalent 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 probability for the equivalence of the concepts in question.

Chapter 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 more at end users and heavily relies on linguistic analysis. It basically makes use of the internal structure of noun phrases to derive ontological knowledge from texts.

The framework by Velardi et al., OntoLearn [VNCN05], 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. In particular, they present a novel algorithm called SSI relying on the structure of the general ontology for this purpose. Furthermore, they include an explanation component for users consisting in a gloss generation component which generates definitions for concepts which were found relevant in a certain domain.

TextToOnto [MS04] 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 [MS00]. The focus of TextToOnto has been so far on the algorithmic backbone with the result that the combination of different algorithms as well as the interaction with the user had been neglected so far. The successor Text2Onto targets exactly these issues by introducing the POM as a container for the results of different algorithms as well as adding probabilities to the learned structures to facilitate the interaction with the user.

Common to all the above mentioned frameworks is some sort of natural language processing to derive features on the basis of which to learn ontological structures. However, all these tools neglect the fact that the document collection can change and that it is unfeasible to start the whole learning process from scratch. Text2Onto overcomes this shortening by storing current results in stores and calculating POM deltas caused by the addition or deletion of documents. Very related is also the approach of [HS98] in which a qualitative calculus is presented which is able to reason on the results of different algorithms, resolving inconsistencies and exploiting synergies. Interesting is the dynamic aspect of the approach, in which the addition of more textual material leads to a reduction in the number of hypothesis maintained in parallel by the system.

Furthermore, as argued in the introduction, all previously developed ontology learning frameworks lack an explanation component helping the user to understand why something has changed in the underlying POM. In addition, most tools do not indicate how certain a learned object actually is, thus making it more difficult for the user to select only the most reliable findings of the system.

As already mentioned, our POMs are not probabilistic in a strict mathematical sense. Several researchers have already addressed the issue of integrating and reasoning with probabilities in knowledge representation formalisms. [DP04] for example present a probabilistic extension of the Ontology Language OWL which relies on Bayesian Networks for reasoning. Other researchers have integrated probabilities into first-order logic [Bac90], the horn fragment of first-order logic [Poo93] or description logics [Jae94, Hei94, KLP97]. Our long-term goal is also to perform probabilistic reasoning on the output POMs in order to rule out inconsistencies in the learned models.

Chapter 6

Conclusion and Future Work

Text2Onto is a framework for data-driven change discovery by incremental ontology learning. It is being developed in SEKT as part of task 3.3. It uses natural language processing and text mining techniques in order to extract an ontology from text and provides support for the adaptation of the ontology over time as documents are added or removed. Explicit modeling of all kinds of changes and an explanation component guarantee maximum transparency and traceability of the ontology learning process.

In future versions of Text2Onto a graphical workflow engine will provide support for the automatic or semi-automatic composition of complex ontology learning workflows. For transforming the POM into a consistent OWL or RDF ontology we aim at a tight integration with the KAON evolution framework (initially created as part of [Sto04], extended in task 'T3.1 Incremental Ontology Evolution' [HSV04]) which will allow to detect and resolve inconsistencies in the generated POMs. The development of the explanation component will be carried on with particular regard to the DILIGENT methodology [PTS04]. By generating machine readable explanations we will make a major step in the direction of making Text2Onto part of the DILIGENT process. We are currently preparing an evaluation setting for comparing the results of the newly developed ontology learning algorithms with previous implementations provided by TextToOnto and other ontology learning tools. Moreover, a detailed user evaluation will offer valuable clues to the usability of the graphical user interface and the benefits gained from the availability of an explanation component.

Bibliography

[AM02] E. Alfonseca and S. Manandhar. Extending a lexical ontology by a combination of distributional semantics signatures. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, 2002.

[Bac90] F. Bacchus. *Representing and Reasoning with Probabilistic Knowledge*. MIT Press, 1990.

[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.

[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.

[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.

[CMT00] H. Cunningham, D. Maynard, and V. Tablan. JAPE: a Java Annotation Patterns Engine (Second Edition). Research Memorandum CS–00–10, Department of Computer Science, University of Sheffield, November 2000.

[CPSTS04] P. Cimiano, A. Pivk, L. Schmidt-Thieme, and S. Staab. Learning taxonomic relations from heterogeneous sources. In *Proceedings of the ECAI 2004 Ontology Learning and Population Workshop*, 2004.

[CV04] P. Cimiano and J. Völker. Towards large-scale, unsupervised and ontology-based named entity recognition. Technical Report. AIFB, University of Karlsruhe, 2004.

[DP04] Z. Ding and Y. Peng. A probabilistic extension to ontology language owl. In *Proceedings of the 37th Hawaii International Conference on System Sciences*, 2004.

[Gen91] M.R. Genesereth. Knowledge interchange format. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, 1991.

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

[Gru93] T.R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[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.

[Hei94] J. Heinsohn. Probabilistic description logics. In *Proceedings of UAI-94*, pages 311–318, 1994.

[HS98] Udo Hahn and Klemens Schnattinger. Towards text knowledge engineering. In *AAAI/IAAI*, pages 524–531, 1998.

[HSV04] P. Haase, Y. Sure, and D. Vrandecic. Ontology mangement and evolution - survey, methods and prototypes. SEKT deliverable 3.1.1, Institute AIFB, University of Karlsruhe, 2004.

[Jae94] M. Jaeger. Probabilistic reasoning in terminological logics. In *Proceedings of KR-94*, pages 305–316, 1994.

[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.

[KLP97] D. Koller, A. Levy, and A. Pfeffer. P-classic: A tractable probabilistic description logic. In *Proceedings of AAAI-97*, pages 390–397, 1997.

[KLW95] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42:741–843, 1995.

[Lee99] Lillian Lee. Measures of distributional similarity. In *37th Annual Meeting of the Association for Computational Linguistics*, pages 25–32, 1999.

[May05] D. Maynard. Sekt deliverable d1.4.1, 2005. SEKT Deliverable.

[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.

[MS04] A. Maedche and S. Staab. Ontology learning. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, pages 173–189. Springer, 2004.

[Poo93] D. Poole. Probabilistic horn abduction and bayesian networks. *Artificial Intelligence*, 64(1):81–129, 1993.

[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.

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

[SEM01] S. Staab, E. Erdmann, and A. Maedche. Engineering ontologies using semantic patterns. In *Proceedings of the IJCAI'01 Workshop on E-Business and Intelligent Web*, 2001.

[Sto04] Ljlijana Stojanovic. *Methods and Tools for Ontology Evolution*. PhD thesis, University of Karlsruhe, 2004.

[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.