D6.6.3 Integration of TextGarden with KAON

Stephan Bloehdorn (UKARL), Blaž Fotuna (JSI), Dunja Mladenić (JSI) and York Sure (UKARL)

with contributions from:
Marko Grobelnik (JSI), Blaž Novak (JSI) and Johanna Völker (UKARL)

Abstract.
EU-IST Integrated Project (IP) IST-2003-506826 SEKT
Deliverable D6.6.3 (WP6)

Bottom-up integration aims at exploring the synergies which the SEKT core technologies offer for (semi-)automatic creation and maintenance of ontologies and for the application of ontology technologies in certain settings. After the initial status report in month 18, this deliverable focuses on integration activities between the KDD and OM core technologies, in particular between the TEXTGARDEN and KAON tool suites. It is part of a series of bottom-up deliverables D6.6.x, each reporting on bilateral integration activities for different pairings of the core technologies.

Keyword list: KAON, Text2Onto, TextGarden, Knowledge Discovery, Ontology Management, Bottom-Up Integration

Document Id.	SEKT/2005/D6.6.1/v1.0
Project	SEKT EU-IST-2003-506826
Date	January 24, 2006
Distribution	public

SEKT Consortium

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Changes

Version	Date	Author	Changes
0.1	02.10.05	Stephan Bloehdorn	document setup
0.2	22.11.05	Stephan Bloehdorn, York Sure	document structure, input to chapters 2 and 3
0.3	26.11.05	Stephan Bloehdorn, Johanna Völker	input to chapter 1,2,3
0.4	14.12.05	Stephan Bloehdorn, Dunja Mladenić	updates to chapter 2,3,4,5
0.5	18.12.05	Blaž Fortuna	updates to chapter 2,3,4
1.0	22.12.05	Stephan Bloehdorn York Sure	version for review
1.1	23.01.06	Stephan Bloehdorn York Sure	final version after review

Executive Summary

Bottom-up integration aims at exploring the synergies which the SEKT core technologies offer for (semi-)automatic creation and maintenance of ontologies and for the application of ontology technologies in certain settings. After the initial status report in month 18 [Bloehdorn et al., 2005], this deliverable focuses on integration activities between the KDD and OM core technologies, in particular between the TEXTGARDEN and KAON tool suites. It is part of a series of bottom-up deliverables D6.6.x, each reporting on bilateral integration activities for different pairings of the core technologies.

This deliverable starts out with giving a short overview over the individual tools developed within the KDD and OM research activities, namely the TEXTGARDEN and KAON tool suites. In the following it sketches four currently envisioned integration scenarios between the KDD and OM areas:

- Scenario 1: Interoperability of Ontology Learning Tools
- Scenario 2: Active Ontology Learning
- Scenario 3: Meta-Learning from heterogenous evidence
- Scenario 4: Ontology-Based Similarities for ML

These descriptions include brief information about the background, targeted functionality and the envisioned benefit to SEKT for each scenario. These scenarios are then described in more detail in the following two chapters, one reporting on the technical and conceptual details of the integration work done within M13–M24 which focuses on scenarios 1 and 2, while the conceptual framework for the integration work within M25–M36 is presented in the next chapter. We summarize and conclude the presented work in the last chapter.

Contents

Chapter 1

Introduction

SEKT aims at developing technologies for Next Generation Knowledge Management that exploit complementary research areas like Knowledge Discovery (KDD), Human Language Technology (HLT) and Ontology/Metadata Management (OM). Specifically, SEKT aims at developing software to:

- semi-automatically learn ontologies and extract metadata (i.e. discover ontology primitives both on the schema and on the instance level),
- evolve and maintain ontologies
- provide knowledge access by means of ontology-based technologies.

One of the main objectives of SEKT is to combine the three core technologies with the aim of achieving synergy effects by looking at similar tasks from different paradigms and enabling new functionalities.

This report is part of the bottom-up integration work performed in workpackage (WP) 6, specifically task 6.6. In the following, we put the work of this deliverable in the context of the SEKT project and outline its content.

1.1 The SEKT Big Picture

Figure 1.1 gives an overview over the SEKT project. The three SEKT core technologies can be found among the 'Research & Development' activities: WP1 explores approaches for Ontology Generation from a knowledge discovery point of view with a strong focus on text mining. The corresponding tool suite, TEXTGARDEN, is mainly developed within WP1. WP2 uses Human Language Technology for Metadata Generation, especially within the framework of the GATE suite. WP3 researches ontology management infrastructures including functionalities for learning, updating and evolving ontologies

over time and develops the KAON infrastructure further with focus on the KAON2 and TEXT2ONTO components. This report is part of the work performed WP 6 on Integration and is naturally related with the technical workpackages and their interaction.

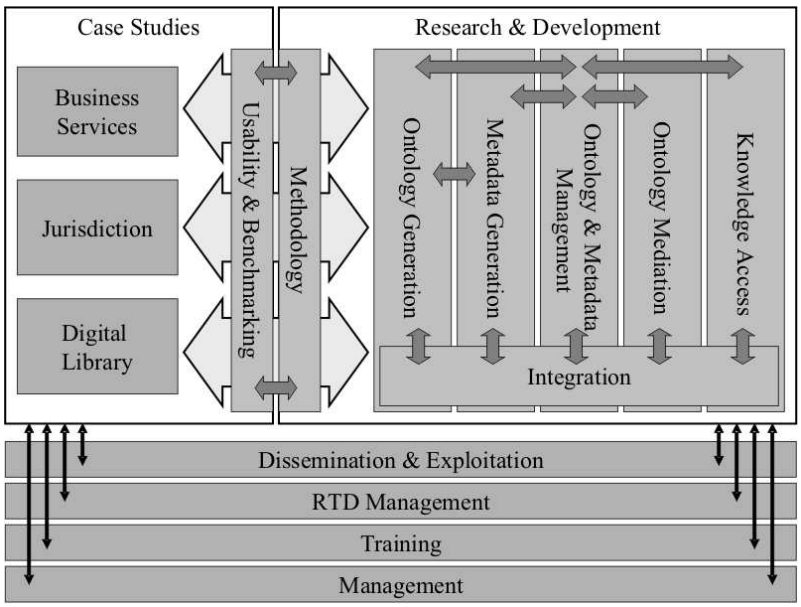

Figure 1.1: The SEKT Big Picture

1.2 Approaches to Integration in SEKT

Workpackage 6 aims to provide integration between the SEKT technical components. This integration is performed in two complementary ways. On the one hand, the *top-down integration activities* pursued within tasks 6.1 – 6.5 aim at a global integration of the individual components via an Integration Plattform. On the other hand, it deals with the complementary approach of *bottom-up integration* on a bilateral basis between the core research partners which is pursued within task 6.6.

1.2.1 Top-Down Integration

In workpackage 6, tasks 6.1 – 6.5 provide the overall architecture framework within which the SEKT technical components will be integrated. In the context of the work done so far it thus deals with the creation and extension of the the SEKT Integration Platform (SIP), which acts as the technical base for top-down integration.

SIP offers modular extensibility for components by wrapping component functionalities in so-called "pipelets" which can be flexibly connected. Further information about the

overall basic platform can be found e.g. in the SEKT deliverables D6.1 – D6.4. Information about the integration of the individual components from the core workpackages into SIP can be found in the 2nd year deliverables D.5.1–D.5.5, which will be complemented by the 3rd year deliverables D.5.6–D.5.10.

The top-down integration allows for bundling of integrated components in many different ways, which is particularly useful within enterprise solutions.

1.2.2 Bottom-Up Integration

Parallel to the top-down integration, SEKT exploits a bottom-up integration strategy. Task 6.6 which deals with bottom-up integration has been included after a first revision of results in task 6.5 at the end of year 1 of the SEKT project. Here, the integration between the core technologies developed in SEKT is performed on a bilateral basis between the three core technical partners.

	KDD	NLP	OM
KDD	WP1 work	**D6.6.2** Integration of ML Approaches in GATE NLP components.	**D6.6.3** Integration of OntoGen and Text2Onto Active Learning for Text2Onto Meta-Learning for Text2Onto
NLP	**D6.6.2** Integration of NLP components in TextGarden.	WP2 work	**D6.6.4** Text2Onto Language Processing based on GATE components.
OM	**D6.6.3** Integration of OntoGen and Text2Onto Ontology Backed Similarity Measures in ML.	**D6.6.4** KAON2 as Backend for GATE Ontology Interface.	WP3 work

Figure 1.2: Bottom-up Integration Activities within SEKT

Bottom-up integration aims at exploring the synergies which the SEKT core technologies offer for (semi-)automatic creation and maintenance of ontologies and for the application of ontology technologies in certain settings. It was felt necessary to better understand the low-level opportunities for integration — first to guide the top-down integration activities, second to emphasize that more research is needed to combine SEKT

technologies. The bottom-up approach complements the top-down strategy in different ways and mainly targets two things:

- Clarify the relationships between knowledge discovery, human language technologies and ontology management.
- Coordinate the efforts for integration of the core technical components delivered in WP1, WP2 and WP3.

In general, the bottom-up integration activities are not meant to replace integration via SIP but aim at examining potential benefits from the combination of core technologies in a light-weight manner. It aims at light-weight proof-of-concept implementations which illustrate the potential and finally evaluate the potential of pairwise integration. This is particularly useful for further research prototyping.

Figure 1.2 gives an overview over the bilateral integration activities pursued within SEKT and the corresponding M24 deliverables within this task. After the initial status report in month 18 [Bloehdorn et al., 2005] which has outlined the overall bottom-up integration efforts from a birds-eye perspective, this deliverable focuses on integration activities between the KDD and OM core technologies, in particular between the TEXTGARDEN and KAON tool suites. It is part of a series of bottom-up deliverables D6.6.x, each reporting on bilateral integration activities for different pairings of the core technologies.

1.3 Outline

This deliverable is structured as follows. In chapter 2, we give a short overview over the individual tools developed within the KDD and OM research activities, i.e. in WP1 and WP3, respectively. In chapter 3 we give a short overview over four currently envisioned integration scenarios between the KDD and OM areas, which includes brief descriptions about the background, targeted functionality and the envisioned benefit to SEKT for each scenario.

In chapter 4 we describe the technical details of the integration work done within M13–M24 which focuses on scenarios 1 and 2, while the conceptual framework for the integration work within M25–M36 is presented in chapter 5, focusing mainly on integration scenarios 3 and 4. We summarize and conclude the presented work in chapter 6.

Chapter 2

Description of the Individual Components for Integration

To make this deliverable self-contained, we shortly describe the tools developed in WP1 and WP3 from the KDD and OM research fields respectively. This chapter is meant as a reference for the next chapters that will refer back to these software tools.

2.1 TextGarden

TEXTGARDEN software tools for text mining is a set of software components, many of them developed mainly within SEKT (workpackage WP1 on Ontology Generation). The objective of this workpackage is to explore various aspects of generating ontological structures by means of machine learning, especially text mining methods. TEXTGARDEN tools enable easy handling of text documents for the purpose of data analysis including automatic model generation and document classification, document clustering, document visualization, dealing with Web documents, crawling the Web and many other. The code is written in C++ and originally runs on Windows platform. TEXTGARDEN consists of a set of command line utilities, which could be run sequentially in pipeline manner to perform a specific learning task. Development of TEXTGARDEN started in 1996 [Mladenić, 1996, Grobelnik and Mladenić, 1998], with major revisions in late 90's. Several components for Text Mining were developed as a part of SEKT WP1 deliverables on the top of the existing library. Here we provide bief description of the software tools emphasizing the SEKT contributed parts where relevant.

2.1.1 TextGarden Library

Functionalities TEXTGARDEN tools enable easy handling of text documents for the purpose of data analysis including:

- Pre-processing of Text Documents in various formats. For SEKT the most relevant format is Bag-Of-Words format with the file extension `.Bow` This format corresponds to the commonly used representation of a text document with a word-vector ignoring position of words in the document.The purpose of the format is to enable efficient execution of algorithms working with the bag-of-words representation such as, clustering, learning, classification, visualization, etc.

- Feature construction components for learning semantic-space of documents that create semantic-space representation of documents based on Latent Semantic Indexing and enables projection of new documents on the learned semantic-space. And another component for feature extraction from images (developed inside D1.3.1)

- Document classification including automatic model generation based on various Machine Learning Algorithms including Support Vector Machines, k-Nearest Neighbour, Logistic Regression, Winnow and more. In SEKT deliverables so far we have used Support Vector Machines (SVM) and k-Nearest Neighbour. This includes also document classification into a large topic ontology (developed inside D1.5.1).

- Processing of unlabelled data and Active Learning (developed inside D1.2.1 [Novak et al., 2004b]). Active learning is implemented on sparse training sets using binary SVM model. It performs active learning loop on the specified input. Semi-Supervised transduction performs a transductive inference on a joint labelled and unlabelled dataset.

- Document Visualization [Fortuna et al., 2005b] Visualization of semantic-space of documents provides visualization of documents as a 2-D map based on the semantic-space representation.

- Focused crawling of the Web (developed inside D1.1.1 [Novak et al., 2004a]) that exploits Google. Additionally there are separate components that enable getting one page from the web based on the Web address specified with URL.

Technical Details TEXTGARDEN consists of a set of command line utilities, which are meant to be combined flexibly in pipeline manner to perform specific learning tasks. The code is written in C++ and originally runs on Windows platforms, a Linux/Unix version of the tools is planned. Using Wine[1] or similar utility TEXTGARDEN can already be run on Linux/Unix platforms. The release of a C++ library for easier integration is planned, too.

[1] `http://www.linux-wine.de/`

2.1.2 TextGarden-OntoGen

ONTOGEN [Fortuna et al., 2005a] is a software tool which helps the user at constructing topic ontologies by the use of different machine learning and text mining methods from TEXTGARDEN. All these methods are seamlessly integrated into a simple graphical user interface (GUI). The ontology construction in ONTOGEN is data-driven and is based on a document collection provided by the user. These documents provide the text mining methods with the information needed for aiding the ontology engineer.

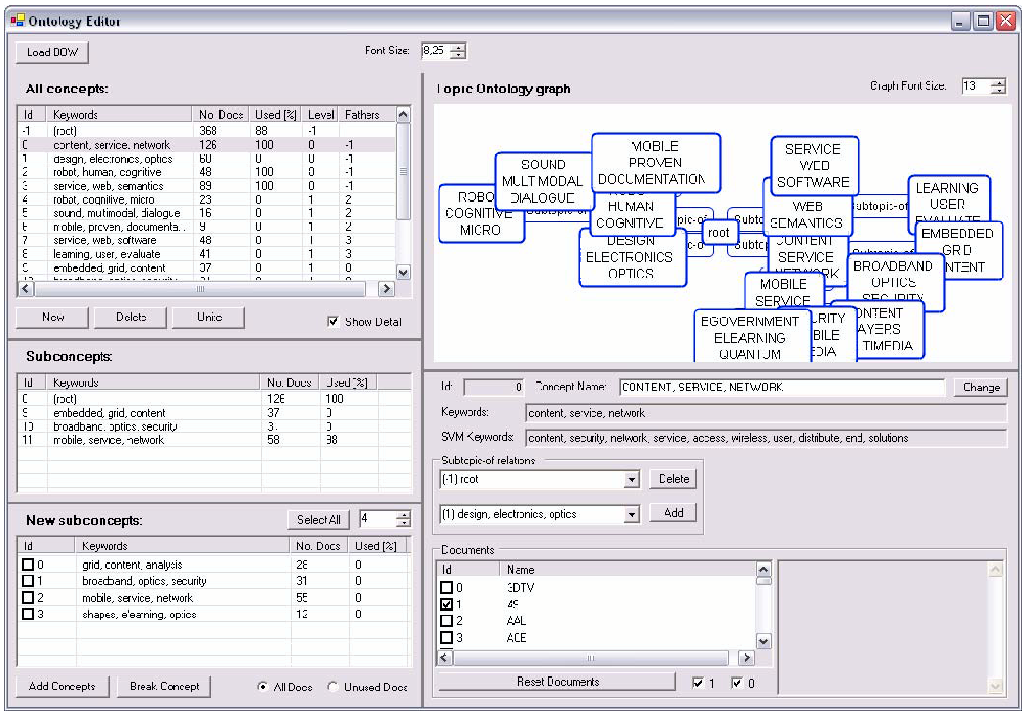

Figure 2.1: Screenshot from OntoGen. See [Fortuna et al., 2005a] for details.

ONTOGEN uses Latent Semantic Indexing (LSI) and k-means clustering for the discovery of topics. Both approaches are well studied and were already successfully used in many applications like information retrieval, cross-lingual text mining, etc and can be very efficiently implemented so they are scalable to the size of the corpora and the dimensionality of the bag-of-words space of the input documents. A linear algebra technique called Singular Value Decomposition (SVD) is used to compute LSI. This method is well studied in areas of numerical linear algebra and efficient algorithms for calculating singular decomposition exist. The basis of LSI is the calculation of singular decomposition of term-document input matrix.

Functionalities The main function of ONTOGEN is the integration of ontology construction methods from TEXTGARDEN and providing a GUI interface to them. The main

functionalities that ONTOGEN offers are:

- suggestion of possible new concepts based on the provided document collection,
- extraction of relevant keywords in order to help naming the concepts,
- detection of outliers for producing cleaner ontologies,
- overview of the ontology by visualizing the concepts their relations,
- help at managing the relations between concepts,
- import and export of topic ontologies as defined in the Proton ontology schema [Terziev et al., 2004] (see section 4.1).

Technical Details The GUI in ONTOGEN is based on .NET framework which works only on Microsoft Windows platform.

2.2 KAON Ontology Management

KAON is a framework of open-source ontology management infrastructure components with focus on scalable and efficient reasoning with ontologies and on learning/evolving ontologies. Two components of the KAON framework are used and developed in the context of SEKT: TEXT2ONTO and KAON2 which will be described in the following.

2.2.1 Text2Onto

TEXT2ONTO [Cimiano and Völker, 2005a] is a complete re-design and re-engineering of our system TEXTTOONTO, a tool suite for learning ontologies from textual data [Maedche and Staab, 2001, Maedche, 2002]. TEXT2ONTO mainly developed from within workpackage 3 to provide functionalities in SEKT for Incremental Ontology Evolution, Usage Tracking for Ontologies and Metadata and especially for Data-driven Change Discovery [Haase and Voelker, 2004].

Functionalities The main functionalities of comprise the learning/discovery of schema-level ontology primitives and instantiations from textual data by analyzing text documents based on linguistic background knowledge and statistical / machine learning approaches. Especially,

- TEXT2ONTO represents the learned knowledge at a meta-level in the form of instantiated modelling primitives within an so called *Preliminary Ontology Model*

(POM). TEXT2ONTO thus remains independent of a concrete target language (like for example OWL or F-Logic) during learning and user supervision phase while being able to translate the instantiated primitives into any (reasonably expressive) knowledge representation formalism, especially into the OWL fragment supported by KAON2.

- user interaction is a core aspect of TEXT2ONTO and the fact that the system calculates a confidence for each learned object allows to design sophisticated visualizations of the POM.
- TEXT2ONTO incorporates strategies for *data-driven change discovery* which allows for consecutive update and evolution of ontologies according to document corpus changes. Besides increasing efficiency in this way, it also allows a user to trace the evolution of the ontology with respect to the changes in the underlying corpus.

In addition to the core functionality of TEXT2ONTO described above we developed a graphical user interface featuring a corpus management component, a workflow editor, configuration dialogues for the algorithms as well as tabular and graph-based POM visualizations. It will be available as an Eclipse[2] plug-in which could facilitate a smooth integration into ontology editors at a later development stage.

Technical Details TEXT2ONTO is written entirely in Java and is thus platform independent. For interaction (import and export) with OWL ontologies, TEXT2ONTO integrates with KAON2.

2.2.2 KAON2

KAON2 is a complete infrastructure for managing OWL-DL and SWRL ontologies. It serves as the main ontology management infrastructure within SEKT and also as the backbone for ontology evolution functionalities within SEKT [Haase et al., 2004]. In the context of this deliverable, KAON2 is introduced mainly for completeness. Integration scenarios 1–3 will work with TEXT2ONTO while scenario 4 will use KAON2 in the background.

Functionalities Main functionalities of KAON2 are:

- a Java API for programmatic management of OWL-DL and SWRL ontologies,
- a stand-alone server providing access to ontologies in a distributed manner,

[2]http://www.eclipse.org

- an inference engine for answering queries,
- a module for extracting ontology instances from relational databases (available soon),
- a query interface for answering SPARQL queries.

The API of KAON2 is capable of manipulating OWL-DL ontologies. Currently, the API can read ontologies in OWL XML Presentation Syntax and in OWL RDF Syntax. For reasoning, KAON 2 supports the $\mathcal{SHIQ}(D)$ subset of OWL-DL (support for datatypes will be available soon). This includes all features of OWL-DL apart from nominals (also known as enumerated classes). Since nominals are not a part of OWL Lite, KAON2 supports all of OWL Lite. The API also provides no direct means for *creating* anonymous individuals. Indirectly, anonymous individuals are however possible by creating random URIs provided they're handled properly (i.e. excluded) in `owl:AllDifferent` and `owl:differentFrom` statements. By means of this mechanism, OWL ontologies that include anonymous individuals can still be processed by KAON2.

KAON2 also supports the so-called DL-safe subset [Motik et al., 2004] of the Semantic Web Rule Language (SWRL). The restriction to the DL-subset has been chosen to make reasoning decidable. Contrary to most currently available DL reasoners, such as FaCT, RACER, DLP or Pellet, KAON2 does not implement the tableaux calculus. Rather, reasoning in KAON2 is implemented by novel algorithms which reduce a $\mathcal{SHIQ}(D)$ knowledge base to a disjunctive datalog program. For an overview of these algorithms, please refer to [Hustadt et al., 2004b]. A detailed (and quite lengthy) technical presentation of all algorithms is given in [Hustadt et al., 2004a].

Technical Details KAON2 is written entirely in Java and is thus platform independent. However, KAON2 requires JDK 1.5 and is not compatible with earlier Java versions.

Chapter 3

Overview on Integration Scenarios

In this chapter, we give a short overview over the integration scenarios that are in the current focus of work at the interface between TEXTGARDEN and KAON. We also place these scenarios in the overall context of the SEKT project.

Scenario 1: Interoperability of Ontology Learning Tools

This scenario targets at the integration between the Ontology Generation Tools ONTOGEN and TEXT2ONTO. We here briefly describe background, targeted functionality and overall benefit to SEKT. The technical details of this integration activity are described in Section 4.1.

Background Within SEKT, two different tools address the Topic of Ontology Generation from different perspectives. On the one hand, the ONTOGEN developed within WP1 [Fortuna et al., 2005a] addresses the problem of topic discovery and semi-automatic construction of topic ontologies. The chosen approach uses LSI and k-means for the discovery of topics. Both approaches are well studied and were already successfully used in many applications like information retrieval, cross-lingual text mining, etc and can be very efficiently implemented so they are scalable to the size of the corpora and the dimensionality of the bag-of-words space.

On the other hand, the TEXT2ONTO software developed within WP3 aims at learning/discovery of schema-level ontology primitives and instantiations from textual data by deeper linguistic analysis and background knowledge.

Target Functionality The first integration scenario aims to prepare the individual tools exchange their data and prepare them for future interoperability within the SEKT ontology engineering plattform. The added value of this integration lies in the complementary nature of the different approaches for the discovery of conceptual structures from text.

While ONTOGEN mainly works on the basis of statistical machinery shallow document representations and as a result is capable of performing computations on very large text databases. On the other hand, TEXT2ONTO addresses the problem of extracting conceptually clean, highly axiomatized ontology primitives by means of natural language processing techniques.

Benefit to SEKT The main benefit of this integration activity lies in the complementary nature of the different approaches for the discovery of conceptual structures from text. We envision uses of both approaches within the SEKT ontology engineering environment. ONTOGEN is likely to be used in scenarios where an initial topic structure is to be derived from large amounts of text documents in an efficient manner, while TEXT2ONTO is likely to complement the derived topic ontologies by means of more formal ontology modelling primitives.

Scenario 2: Active Ontology Learning

This scenario targets at the integration of active learning technology into the TEXT2ONTO framework. The technical and conceptual details of this integration activity are described in Section 4.2.

Background Current components of TEXT2ONTO focus solely on completely unsupervised techniques for detecting ontological primitives within textual data. A major reason for this decision has typically been the high cost and inflexibility that stems for the labelling work that would be required for supervised learning approaches. On the other hand, certain ontology learning tasks like for example the discovery of concept instantiations [Bloehdorn et al., 2005, Appendix A] are very well suited for supervised learning approaches. Active learning algorithms [Novak et al., 2004b] take training examples with only minimal initial labelling and then ask the user to give additional information on selected, particularly informative, training examples.

Target Functionality The integration activity described in this scenario targets at the integration of the active learning technology developed within WP1 [Novak et al., 2004b] into the TEXT2ONTO framework. Currently, this approach is pursued for learning concept instantiations but might be extended to other ontology learning tasks. The combination of existing approaches and this new paradigm also reflects the fact that different ontology learning tasks require different approaches for the discovery of the corresponding primitives.

Benefit to SEKT Integrating supervised learning approaches for TEXT2ONTO will allow TEXT2ONTO to work in settings where only little background knowledge, e.g. about

the linguistic patterns, is available. This will for example allow the faster adaption of components for languages for which no or only minimal patterns are available as for example the Spanish Legal case study within SEKT.

Scenario 3: Meta-Learning from heterogenous evidence

Similar to scenario 2, scenario 3 aims at integrating some supervised machine learning functionality within TEXT2ONTO. The technical details of this activity will be described in section 5.1.

Background TEXT2ONTO currently comes with a number of analysis algorithms for different ontology learning tasks. It is often desirable to combine the results of different analysis algorithms. This combination is currently performed by simple operators like averaging the individual evidence scores or taking the maximum among several of these.

Target Functionality The target functionality is to tune the combination of evidences produced by different analysis modules within TEXT2ONTO by means of regression and/or classification functions which will be estimated given some feedback of the user about the quality of the returned results.

Benefit to SEKT Similar to scenario 2, this functionality will enable an increased adaptivity of TEXT2ONTO algorithms in new domains and (or) languages.

Scenario 4: Ontology-Based Similarities for ML

On the contrary to the previous scenarios, scenario 4 aims at the integration of ontology management technology into the machine learning field without aiming primarily at solving ontology learning tasks. The technical details of this activity will be described in section 5.2.

Background At different stages of the evolution of an ontology, the ontology also reflects valuable information on the similarity of objects based on the current model. This similarity forms a natural input for machine learning algorithms that work on data that caries semantic information, like e.g. annotated documents or users whose usage profiles are stored in the ontology.

Target Functionality WP3 will develop facilities for similarity calculations within the ontology, namely between individual ontology and metadata entities and between complex (aggregated) descriptions based on ontology primitives. Within scenario 4, these

"semantic" similarities will be used as inputs for machine learning algorithms of any kind.

Benefit to SEKT The activities pursued within scenario 4 aim at explicitly exploiting the power of semantic descriptions based on ontological primitives in machine learning paradigms. The resulting similarity measures are a valuable ingredient for applications like for example the FAQ retrieval in the legal case study and recommendations in the BT digital library case study.

Chapter 4

Report on Current Tool Integration

This chapter describes in more technical detail those scenarios that have already achieved a high level of technical integration beyond the conceptual framework. This applies for integration scenarios 1 and 2, which will be described in section 4.1 and section 4.2 respectively.

4.1 Scenario 1: Interoperability of Ontology Learning Tools

4.1.1 Motivation: Interoperability

This scenario targets at the integration between the Ontology Generation Tools ONTOGEN and TEXT2ONTO.

Two different tools, namely ONTOGEN and TEXT2ONTO, are developed and maintained within SEKT that address the Topic of Ontology Generation from two very different perspectives. This integration scenario aims at making the individual ontology learning tools interoperable and exchange their data. The added value of this integration lies in the complementary nature of the different approaches for the discovery of conceptual structures from text. On the one hand, the ONTOGEN software developed within WP1 [Fortuna et al., 2005a] tries to address the issues of topic discovery and semi-automatic construction of topic ontologies as outlined in section 2.1.2 by means of LSI and k-means for the discovery of topics based on comparatively shallow document representations. On the other hand, the TEXT2ONTO software developed within WP3 and described in section 2.2.1 aims at the discovery and induction of schema-level ontology primitives and instantiations from textual data, typically by deep linguistic analysis and background knowledge.

4.1.2 Implementation within OntoGen

Within this integration work, the functionality of saving and loading topic ontologies from files in RDF format that can be processed within the KAON tool suite has been implemented within ONTOGEN. Topic ontologies created with ONTOGEN can be saved as RDF files so it can be used in other tools and ontologies created with other ontology engineering tools, e.g. an export from TEXT2ONTO, can be opened with ONTOGEN for additional editing or refining.

The basic schema used is the Proton Ontology developed within WP1 [Terziev et al., 2004]. The skeleton of topic ontology is stored using following concepts and relations:

- `http://proton.semanticweb.org/2005/04/protont#Topic`,
- `http://proton.semanticweb.org/2005/04/protont#Document`,
- `http://proton.semanticweb.org/2005/04/protont#subTopic`,
- `http://proton.semanticweb.org/2005/04/protont#hasSubject`.

The topics' names and the documents' abstracts are stored using:

- `http://proton.semanticweb.org/2005/04/protons#description`,
- `http://proton.semanticweb.org/2005/04/protont#documentAbstract`.

TEXTGARDEN can read topic ontologies stored using this schema. RDF exports created with ONTOGEN all follow this schema.

In order to avoid loosing application specific data (mainly on the generation process) when exporting ontologies in ONTOGEN we have extended Proton[1] with the following two concepts:

- `http://kt.ijs.si/blazf/jsikm#OntoGenClassProperties`,
- `http://kt.ijs.si/blazf/jsikm#OntoGenInstanceProperties`.

These two concepts allow ONTOGEN to store the extra information about topics or documents such as keywords of topics and locations of documents. These two concepts are connected to Topic and Document concepts from Proton with relations:

- `http://kt.ijs.si/blazf/jsikm#hasOntoGenClassProperties`,

[1] While the namespace for these extensions has not been agreed upon, the information is likely to reside within the JSI namespace.

- `http://kt.ijs.si/blazf/jsikm#hasOntoGenInstanceProperties`.

By using these extensions no information regarding the topic ontology constructed in ONTOGEN is lost when saved as RDF. However, they are only optional and are not necessary condition for opening RDF file in ONTOGEN.

For better integration with BT Digital library case study, ONTOGEN also stores the following relations:

- each topic has to reference its documents with `diglib#hasArticle` (in topic ontology as defined in Proton only documents have links to topics),
- titles of documents are stored using `diglib#title`.

ONTOGEN uses these relations only when exporting to RDF files. They are not used at importing topic ontologies from RDF files into ONTOGEN. Similarly, all other relations besides the ones mentioned here are ignored when reading RDF files. Figures 4.1 and 4.2 show code fragments for the topic and document related statements respectively.

```
<ptop:Topic rdf:about="#TOP_166">
   <psys:description>Loans</psys:description>
   <diglib:hasArticle rdf:resource="#DOC_84" />
   ...
   <diglib:hasArticle rdf:resource="#DOC_5043" />
   <ptop:subTopicOf rdf:resource="#TOP_26" />
   <jsikm:hasOntoGenClassProperties rdf:resource="#CLS_PROP_166" />
</ptop:Topic>

<jsikm:OntoGenClassProperties rdf:about="#CLS_PROP_166">
   <jsikm:hasCentroidKeywords>
      banking, loans, investment
   </jsikm:hasCentroidKeywords>
   <jsikm:hasSVMKeywords>
      million, loans, security
   </jsikm:hasSVMKeywords>
</jsikm:OntoGenClassProperties>
```

Figure 4.1: Code Fragment from ONTOGEN Proton/RDF export for a saved topic.

4.1.3 Outlook

The current integration has mainly focused on the technical issue of allowing ONTOGEN to export and import ontologies in RDF/OWL format. Future integration of these two

```
<ptop:Document rdf:about="#DOC_84">
   <ptop:hasSubject rdf:resource="#TOP_0"/>
   <ptop:hasSubject rdf:resource="#TOP_26"/>
   <ptop:hasSubject rdf:resource="#TOP_166"/>
   <jsikm:hasOntoGenInstanceProperties rdf:resource="#INST_PROP_84"/>
   <diglib:title>WRO</diglib:title>
   <ptop:documentAbstract>
      Woronoco Bancorp, Inc. (the Corporation) has no significant
      assets other than all of the outstanding shares of Woronoco
      Savings Bank (the Bank). The Bank's business consists of the
      acceptance of retail deposits...
   </ptop:documentAbstract>
</ptop:Document>

<jsikm:OntoGenInstanceProperties rdf:about="#INST_PROP_84">
   <jsikm:locationOfInstance>docs.bow#84</jsikm:locationOfInstance>
</jsikm:OntoGenInstanceProperties>
```

Figure 4.2: Code Fragment from ONTOGEN Proton/RDF export for a saved document.

complementary ontology engineering components could be envisioned within the SEKT ontology engineering platform, ONTOSTUDIO.

4.2 Scenario 2: Active Ontology Learning

4.2.1 Motivation: Instance Classification

In Ontology Learning and Semantic Annotation, the task of instance classification can be shortly described as assigning instances or named entities appearing in the corpus to their correct concept in the ontology.

Building upon the formal descriptions of ontologies as introduced in deliverable D.6.6.1 [Bloehdorn et al., 2005][2], we can formalize the instance classification task as the learning of concept instantiations.

Definition 1 (Concept Instantiation) *The* extension $[\![c]\!] \subseteq I$ of a concept $c \in C$ *is re-*

[2]This formalization mainly aimed at providing mathematically strict definitions of ontological modelling primitives which are present in almost all ontology definition languages. These definitions, were mainly meant to study structural aspects important for ontology learning tasks without committing too much to a specific ontology representation language. Please refer to [Bloehdorn et al., 2005] for details on this formalization.

cursively defined by the following rules:

- $[\![c]\!] \leftarrow \iota_C(c)$
- $[\![c]\!] \leftarrow [\![c]\!] \cup [\![c']\!]$, *for* $c' <_C c$.

where $\iota_C(c)$ *corresponds to the function which returns the set of instances that are explicitly stated in the ontology to instantiate a certain concept* $c \in C$.

Typically, we distinguish *Instance Extraction* from *Instance Classification* as follows:

Definition 2 (Instance Extraction) *The instance extraction task aims at the identification of potential instances* $[\![c]\!]$ *of concepts* $c \in C$.

Definition 3 (Instance Classification) *Given a set of instances that are to be added to the ontology, the instance classification (or concept instantiation) task is to discover which concept these instances actually instantiate. In terms of the ontology model this means learning instantiations* $[\![c]\!]$ *for* $[\![c]\!] \subseteq I$ *and* $c \in C$. *For the metamodel this means learning* instantiates *relationships between instantiations of* Instance *and* Concept.

Up to now, the TEXT2ONTO system (see section 2.2.1 and [Cimiano and Völker, 2005a]) employed two groups of unsupervised approaches to instance classification:

1. Unsupervised Instance Classification as a Retrieval Problem
2. Unsupervised Instance Classification by means of linguistic patterns

The first approach relies on a fully unsupervised similarity-based technique for 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 retrieval problems. As similarity measure TEXT2ONTO uses the Skewed divergence as it was found to perform best in experiments. Using this similarity measure as well as further heuristics, TEXT2ONTO achieved an F-Measure of 32.6% when classifying instances with respect to an ontology comprising 682 concepts [Cimiano and Völker, 2005b]. The approach is unsupervised as it relies on no labelled training data and is open-domain as the ontology can simply be exchanged.

The second approach for instance classification in TEXT2ONTO relies on matching a library of linguistic patterns that are likely to indicate concept instantiations as e.g. in *"...cities such as Paris..."*, where a concept instantiation between the concept city and the instance Paris can be inferred.

The reason to rely on unsupervised approaches which do not require pre-labelled input data has the advantage of being open-domain in the sense that the underlying ontology and the corpus can be replaced. This can easily be accomplished if one resorts to an unsupervised system since providing labelled training data for a few hundred concepts as we consider in our approach is often unfeasible.

At the same time, however, the presented approaches also come with an inherent deficiency each:

1. For the first approach, a global similarity measure may be too crisp for the many different types of instance-concept pairing observed in natural language. In some cases it may be also rare that an instance and the corresponding concept actually occur in a similar context.

2. For the second approach, while being highly reliable, it may be cumbersome to design reliable language patterns, be it for specific types of concepts or even for totally different languages than English, e.g. for the Spanish language as required in the SEKT legal case study [Casanovas et al., 2004].

On the other hand, *supervised approaches* that try to induce an accurate predictive model based on an analysis of the underlying statistical patterns *for a specific classification task* have shown success in many real-world problems amongst them for instance or named entity classification tasks and are flexible in adapting to very different situations. However the knowledge acquisition bottleneck in form of the requirement of large amounts of pre labelled data remains.

Dealing with unlabelled data in supervised scenarios has been one of the objectives of the work in WP1. The active learning algorithms described in [Novak et al., 2004b] are well-suited for the situation of instance classification where a small number of initial high confidence labels (e.g. generated by the second unsupervised approach presented above) are available but this data isn't enough to train supervised classification models right away. Active Learning algorithms take training examples with only minimal initial labelling and then interactively ask the user to give additional information on only few selected, particularly informative, training examples. We describe the active learning paradigm in the following section.

4.2.2 Active Learning

As pointed out in [Novak et al., 2004b] and [Novak et al., 2005] active learning has a strong link to the problem of 'experiment design' addressed in statistical literature. It is a generic term describing a special, interactive kind of a learning process. In contrast to the usual (passive) learning where the student is presented with a static set of examples that are then used to construct a model, active learning paradigm means the student can

"ask" the "oracle" (e.g. domain expert, user,...) for a label of an example. Figure 4.4 illustrates the situation.

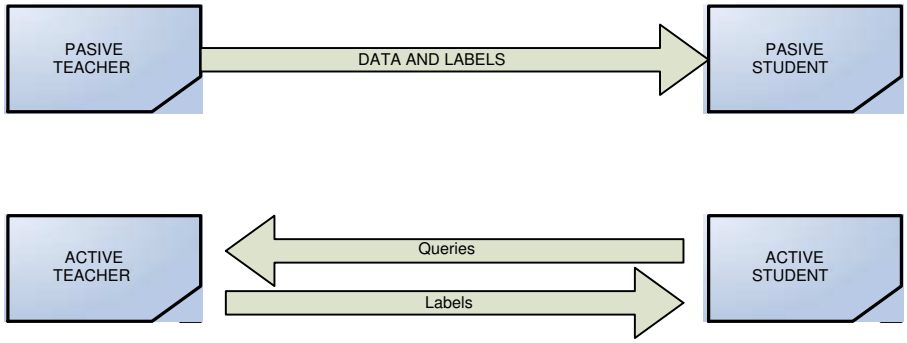

Figure 4.3: Illustration of passive versus active learning paradigms [Novak et al., 2004b].

Based on the published results TEXTGARDEN implements the *Simple Margin method* described in [Tong and Koller, 2000], active learning with sampling estimation of error reduction. All of the methods are provided with two input pools of labelled and unlabelled feature vectors. In each iteration they are allowed to return up to a predefined constant number of queries (indices of unlabelled instances). With each query a decimal number signifying the estimated importance of the query is also submitted. The Simple Margin algorithm creates a linear SVM based on currently labelled examples in each iteration of the loop. It then calculates the distance of all of the unlabelled samples from the hyperplane, orders them by ascending order and selects first N to be sent for user labelling.

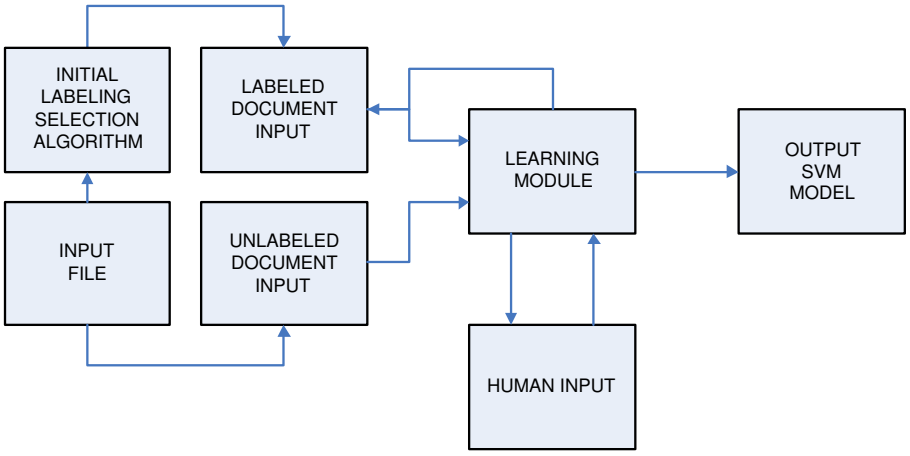

Figure 4.4: Active learning architecture as implemented in TEXTGARDEN [Novak et al., 2004b].

Because of the inherent interactive nature of active learning, the active learning binary of TEXTGARDEN uses the standard input/output for communication with the user. The

communication protocol is line based and asynchronous. Every line written to standard output is a query and has the form

```
QUERY-ID ``MEM'' INSTANCE-ID
```
or
```
QUERY-ID ``IS-A'' INSTANCE-ID CURRENT-MARGIN CATEGORY-ID
```

and the answer for both query types read from standard input has a form of

```
QUERY-ID CATEGORY-ID
```

or an "END" literal. Here, QUERY-ID is an ASCII representation of an unsigned integer used for associating answers with queries, INSTANCE-ID is the TEXTGARDEN ID of an instance, CATEGORY-ID likewise for categories, i.e. positive or negative with respect to a concept in question. The "END" string instructs the program to finish immediately and save the current model and results - even if not all of the input samples were queried for.

4.2.3 Prototype Implementation for Text2Onto

The integration activity within scenario 2 aimed at providing TEXT2ONTO with supervised instance classification functionalities without requiring large amounts of hand-labelled input data. The TEXTGARDEN active learning component was chosen to enable this functionality. Here, learning the concept instantiations of a given class is formalized as a supervised learning problem. Obviously, this design decision requires the ontology engineer working with TEXT2ONTO to undergo a "training session" for each concept in question during which she is required to give answers to questions of the type *"Is instance x an instantiation of concept y?"*. During this interaction, a more and more accurate classification model is built. Depending on the achieved quality and the total number of questions, the system may end the training process itself or the user can force the end of the training process whenever desired. In the following, we summarize the consecutive steps that the system follows within the instance classification task

Instance and Concept Extraction: Initial Extraction of potential concepts and instances is done by the current TEXT2ONTO algorithms available for the purpose. As usual, the corresponding primitives are stored as part of the Preliminary Ontology Model (POM).

Initial Labelling: The initial labelling of the instances with respect to the extracted concepts is done by the usual TEXT2ONTO algorithms, e.g. using the linguistic "Hearst"-patterns which is stored (with corresponding confidence values) in the POM.

Construction of Learning Problem: After the user has initiated the active learning module and chosen the concept for which instantiations should be discovered, the initial

instantiations are checked for positive or negative examples. Using a simple heuristic, those instances that were seen as instantiations of the target concept with a confidence value above 0.9 are regarded as positive training examples while those with a confidence value below 0.1 are regarded as negative examples. All other discovered instances are added to the learning problem as unlabelled examples. The user is presented an dialog which summarizes the learning problem and allows him to make changes according to his view on the problem (see figure 4.5).

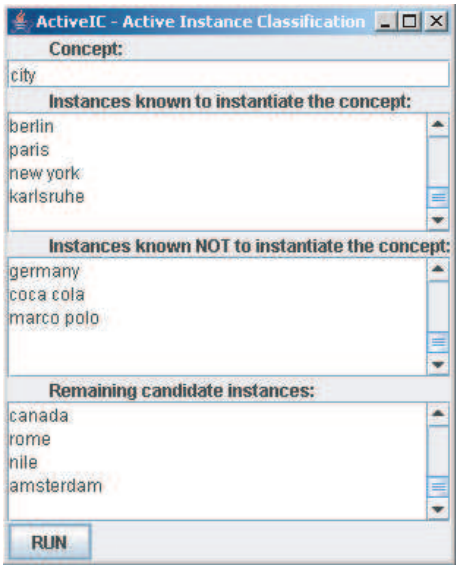

Figure 4.5: Prototype Screenshot of Active Learning Instance Classification Module - Start Configuration.

Feature Extraction: For each instance that is part of the learning problem, context feature vectors are extracted from the sentence the respective instance occurs in. Here, each occurrence of a word in the same sentence as the target instance is seen as a positive count for the respective word feature.

Active Learning Loop: After the learning problem and feature vectors have been created, the active learning loop is started. Internally, the learning problem is saved as an TEXTGARDEN learning problem and the TEXTGARDEN learning component is called with the file of the learning problem as parameter. The `STDOUT` and `STDIN` communication of the TEXTGARDEN active learning component concerning questions to the user on the labelling of certain instances is propagated to the GUI for user feedback (see figure 4.6):

Export: In the current prototype implementation, the result of the active learning loop is exported into a separate file[3]. In a future version, these are, however, meant to be

[3]Note that the results are necessarily consistent with the POM as only instances with unknown target concepts have received new labels.

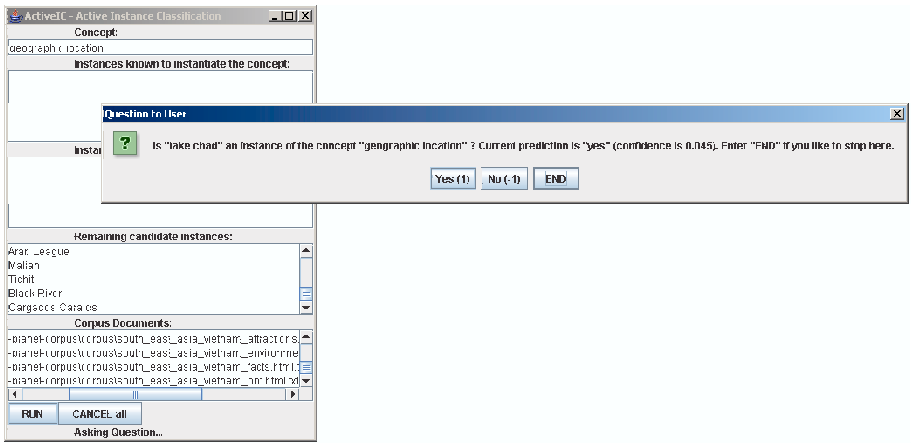

Figure 4.6: Prototype Screenshot of Active Learning Instance Classification Module - Question Dialog.

written back to the POM directly.

4.2.4 Outlook

This integration activity has mainly focused on exploring a principled way of using active learning for instance extraction. The available prototype implementation will be experimentally evaluated on data available from the case studies. Based on these results, it will be refined and extended with respect to the following aspects. Firstly, future work will aim at evaluating the unsupervised and active-learning-based approaches to instance extraction within TEXT2ONTO with respect to different target concepts, different contextual features and user interaction. We will use these results to tune and refine the integration. Secondly, a different route for future work can be seen in incorporating further refinements beyond the simple margin method into the active learning algorithm. A third cluster of future work can be seen in the exploration of active learning technology for other tasks than for instance classification as e.g. for relation instantiation. Finally, we aim at bringing these components to work within the case study scenarios.

Chapter 5

Report on Scheduled Tool Integration

This chapter describes in more technical detail those scenarios for which the actual implementation work is scheduled for M25–M36, for which, however, the conceptual framework has already been developed. This applies for integration scenarios 3 and 4, which will be described in section 5.1 and section 5.2 respectively.

5.1 Scenario 3: Meta-Learning from heterogenous evidence

Similar to scenario 2, this scenario (scenario 3) aims at integrating some of the supervised machine learning functionality available in TEXTGARDEN within TEXT2ONTO, however in a slightly different direction as presented in the following.

5.1.1 Motivation: combining heterogenous evidence

TEXT2ONTO currently comes with a number of analysis algorithms for different ontology learning tasks. Each different algorithm is able to attach a level of confidence within the range $[0, 1]$ to the primitives it has detected. In many learning tasks, multiple different learning algorithms are available for learning the respective primitives of the task. The results of these individual algorithms can be seen as discovering *evidences* for a certain ontology modelling primitive.

Currently, the combination of heterogenous evidences in TEXT2ONTO for a certain ontology modelling primitive is done by means of comparatively simple mathematical operators like *average, maximum, minimum* and the like on the individual confidence scores.

Example 1 *Algorithm A has detected a* `subclass-of` *relationship between entities*

E_1 and E_2 with a (low) confidence score of 0.2 while algorithm B is very confident on the `subclass-of` *relationship and outputs a confidence score of 0.9. The* `average combiner` *would conclude an overall confidence on a* `subclass-of` *relationship between entities E_1 and E_2 of 0.55.*

Obviously, such simple combinations of evidences may be likely to smooth the predictions but they are not likely to discovery the best approximation to the unknown correctness of the respective primitive.

In scenario 3, we aim at learning, i.e. estimating parameters for more complex combiners. In this sense, we regard the output of the individual learning algorithms as input to a supervised meta-level learning algorithm for TEXTGARDEN.

Example 2 (Linear Combination Function) *Given numeric inputs from numeric inputs from n different base learning algorithms, i.e. a n-dimensional vector x, the learning problem for the combiner on the meta-level corresponds to learning a linear function f on the input data:*

$$f : \mathbb{R}^n \rightarrow \mathbb{R}$$
$$f(x) = \langle x, w \rangle + b = \left(\sum_{i=1}^{n} x_i w_i \right) + b.$$

The variables w and b correspond to the weight- and bias-parameters of the approximation function which need to be discovered.

Obviously, this approach requires the availability of input data, i.e. instances of the respective primitive to be learned, confidences of the base learners as input for the learning process and human judgements on the correctness of the primitives in question. Depending on the question whether the user supplies crisp labels (i.e. yes/no) or also confidence values in a range $[0, 1]$ as target values, this problem corresponds to a *classification* or *regression* problem respectively which can be performed easy and efficiently based on the regression and classification algorithms available in TEXTGARDEN.

The envisioned use case is that optimal parameters (as e.g. weights in case of a linear combination function) are learned only when new base learning algorithms are available requiring extensive user interaction to supply the results. The tuned parameters can then be saved as a specific instantiation of a combiner module for future use.

5.1.2 Implementation Plan and Applications within SEKT

This scenario will be evaluated experimentally within months 25—36. Based on the results of this evaluation, the actual implementation within TEXT2ONTO using TEXTGARDEN modules will be initiated.

Similar to scenario 2, this functionality will enable an increased adaptivity of TEXT2ONTO algorithms in new domains and specially to languages other than English as for example Spanish in the legal case study.

5.2 Scenario 4: Ontology-Based Similarities for ML

While, the previous sections have mainly addressed integration scenarios that aim at improving ontology learning tasks, especially by means of integrating machine learning technology into ontology management, scenario 4 tries to address the opposite direction.

5.2.1 Motivation: Data Representation in Machine Learning

Most of current day's machine learning algorithms use the notion of simple feature vectors in learning to cluster, classify and rank data items.

The data items which form the input to the knowledge discovery algorithm could be texts that should be classified into topics or users that should be ranked with respect to to their similarity to other users as for example in collaborative filtering scenarios. In these and in other scenarios, the data items are represented as flat vectors, each dimension of which corresponds to a certain characteristic, called a "feature" that is attributed to the data item in question and typically takes values from a range of numeric values.

Example 3 (Bag-of-Words) *In the Bag-of-Words representation of text documents there is a dimension for each word contained in the document. A document is then encoded as a feature vector with word frequencies as elements. The frequencies are typically weighted by some further weighting scheme, for example the Term-Frequency-Inverse-Document-Frequency (TFIDF) weighting scheme, where all i-th elements are multiplied with* $IDF_i = \log(N/df_i)$*, where* N *is total number of documents and* df_i *is the number of documents in which i-th word appears.*

Example 4 (Collaborative Filtering) *In a typical collaborative filtering scenario, users are represented as vectors, where the each dimension of the vector corresponds to a certain "product" and the value this dimension takes, corresponds to the number of "purchases" of that product. While typically phrased "purchases" and "products" for historical reasons, the recorded incidents could be far more general, for example in the case when the "product" corresponds to a certain website and the "purchase" actually corresponds to a single visit to that website.*

These representations are often referred to as the *vector-space model*, a term used which is especially used in the information retrieval and text mining communities [Salton and McGill, 1983]. The basic idea behind these vector models is to enable a

straight-forward calculation of similarities between the data items which is typically implemented as a similarity-function sim on the vectors:

$$sim : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}.$$

Typical choices of the similarity functions are the *dot product* between vectors, or more frequently the dot product between normalized vectors, i.e. the *cosine measure*.

Example 5 (Cosine Measure) *The cosine similarity measure between two vectors is given by the formula:*

$$sim_{cosine}(x, y) = \frac{\langle x, y \rangle}{\|x\|_2 \cdot \|y\|_2}.$$

Retrieval and machine learning tasks are thus based on the vector model and corresponding similarities. In retrieval tasks, "similar" data items are seen to correspond to items sharing a high similarity according to the chosen similarity function. In classification tasks, the similarities are often required to correspond to variations of the dot-product (as in the cosine similarity example) of the feature vectors involved – the classification model is then built based on the geometry implied by the pairwise similarities.

Experience shows that this approach is simple, easy to implement and successful in many diverse application scenarios. However, despite its success in many scenarios, the plain vector space model relies on an assumption which is not valid in many real-world tasks: *The characteristics of the data items are seen as independent from each other as they are mapped to independent vector space dimensions*.

However, in many real world settings, this assumption is only partly true, as for example sometimes in the bag-of-words model.

Example 6 (Problems Bag-of-Words) *By representing documents as vectors within the space of single words, the chosen learning or retrieval algorithms are restricted to detecting patterns and similarities in the used* terminology *only, while* conceptual *patterns remain ignored. Specifically, systems using only words as features exhibit a number of inherent deficiencies:*

1. Multi-Word Expressions *with an own meaning like* "Semantic Web" *are chunked into pieces with possibly very different meanings like* "web".

2. Synonymous Words *like* "tungsten" *and* "wolfram" *are mapped into different features.*

3. Polysemous Words *are treated as one single feature while they may actually have multiple distinct meanings.*

4. Lack of Generalization*: there is no way to generalize similar terms like "beef" and "pork" to their common hypernym "meat".*

While items 1 – 3 of the previous example directly address issues that arise on the lexical level, which may be tackled by appropriate linguistic preprocessing item 4 rather addresses an issue that is situated on the conceptual level.

Example 7 (Problems of Flat Vectors in Collaborative Filtering) *By representing the different users as vectors of individual purchase incidents, the chosen learning or retrieval algorithms are restricted to detecting patterns and similarities within the actual purchases only, while patterns on the level of product groups remain ignored. For example, a digital library user who visited a distinct article is – as such – not similar to other users who didn't visit the very same article but maybe other articles that belong to the same conceptual group, e.g. articles in "ontology learning".*

Of course, research in KDD and machine learning has already tried to address this kind of problems. Approaches for dealing with redundant and/or interrelated features without explicit background information have been proposed from the field of *dimension reduction*, as for example by means of Singular Value Decomposition (SVD) or Principal Component Analysis (PCA) among others. The main trait these approaches share is to discover redundant information within the full set of data items and their corresponding feature vectors, i.e. in the full "data $\times$ feature matrix" by means of techniques from statistics or linear algebra and project the original information into a lower dimensional "semantic" vector space.

Example 8 (LSI) *Latent Semantic Indexing (LSI) is a technique for implicitely extracting semantic background knowledge from text documents. It uses a technique from linear algebra called Singular Value Decomposition (SVD) for extracting words with similar meanings. This can also be viewed as extraction of hidden semantic concepts or topics from text documents. First, the term $\times$ document matrix T is constructed from a given set of text documents. This matrix is decomposed using singular value decomposition such that $T = U\Sigma V$ where the matrices U and V are orthogonal and contain the left and right eigenvalues, respectively while Σ is a diagonal matrix with ordered singular values on the diagonal. The columns of the matrix U form an orthogonal basis of a subspace in bag-of-words space where vectors with higher singular values carry more information. Because of all this, vectors that form the basis can also be viewed as "latent" concepts or topics. The LSI approach aims at choosing only a subset of k most informative singular values and cutting of the remaining dimensions. Learning algorithms then work in a lower-dimensional "semantic" space.*

While these approaches work without any human intervention, the question how to choose the number of remaining dimensions k as in the LSI example and which dimension

reduction technique to use brings up a new dimension of complexity. Also, these models tend to show a "black-box" behavior, where the user is confronted with accepting the calculated interpretation of "semantics" without actually understanding the fine grained interdependencies.

On the other hand, when conceptual background knowledge is available in the form of ontologies and metadata annotations a different direction introducing higher-level semantics can be pursued which will be the core of integration scenario 4.

5.2.2 Ontology-based Similarities for Data Items

We can look at the data items that form the input to the learning algorithm as instances of an ontology and their corresponding features as the concepts they instantiate. At different stages of the evolution of an ontology, the ontology also reflects valuable information on the similarity of objects based on the current model. This similarity forms a natural input for machine learning algorithms that work on data that caries semantic information, like e.g. annotated documents or users whose usage profiles are stored in the ontology. Building again upon the formal descriptions of ontologies as introduced in deliverable D.6.6.1 [Bloehdorn et al., 2005], we can formalize this notion in a similar fashion as we did in section 4.2:

Definition 4 (Concept Instantiation) *The* extension $[\![c]\!] \subseteq I$ of a concept $c \in C$ *is recursively defined by the following rules:*

- $[\![c]\!] \leftarrow \iota_C(c)$
- $[\![c]\!] \leftarrow [\![c]\!] \cup [\![c']\!]$, *for* $c' <_C c$.

Correspondingly, the set of concepts $con(i)$ *for an instance* $i \in I$ *is recursively defined by the following rules:*

- $con(i) \leftarrow \iota'_C(i)$
- $con(i) \leftarrow con(i) \cup \{c' \in C | c' >_C c, c \in con(i)\}$.

Here, we have used the function ι'_C to denote the *direct conceptualization* function as defined in the following.

Definition 5 (Instance Conceptualization)

$$\iota'_C : I \rightarrow \mathfrak{P}(C)$$

$$\iota'_C(i) = \{c \in C | i \in \iota_C(c)\}$$

This view allows to introduce new types of similarities and/or data representations which can be used to allow for detecting more complex and accurate patterns within data. This approach is flexible and versatile as long as a set of features can be found that can be organized in a hierarchical (taxonomic) way.

Definition 6 (Topic Ontologies in a Recommendation Scenario) *In a recommendation scenario users may be described by sets of topics they are interested in. If the topics, as for example the InSpec topics used in the BT digital library case study, are arranged in a taxonomic manner, the taxonomy may implicitely let two users share a common supertopic, although their explicit topic profiles might be distinct.*

There are two different, though related, approaches for exploiting the hierarchical structure of the features:

1. Explicitly compiling higher level features directly into the feature representations for the individual data items.
2. Using the taxonomic structure of the features implicitly within the evaluation of the pairwise similarities.

Example 9 (Explicit Feature Expansion in the Bag-of-Words Model) *In the bag-of-words model, explicit feature expansion could be used by compiling hypernyms for each word that is explicitly encountered in the text into the feature vector. Typically, this is done up to a certain level in the taxonomy or up to a certain distance from the source features. See for example [Bloehdorn and Hotho, 2004, Bloehdorn et al., 2006, Scott and Matwin, 1998]*

The 2nd approach is exemplified by the so called "semantic" kernel, first introduced in [Siolas and d'Alché Buc, 2000]. At this point we do not introduce the theory behind kernels in detail. In the context of this deliverable it is sufficient to characterize a kernel function as any similarity function between two inputs for which an interpretation as a dot product in a some vector space into which the data could theoretically be mapped can be given. The interested reader is referred to [Shawe-Taylor and Cristianini, 2004] for further information.

Example 10 (Affine "Semantic" Kernel for Text Classification) *In this approach, the information about the similarities across individual dimensions is coded in a proximity matrix P and exploited in the evaluation of the dot product between two vectors:*

$$K_{affine}(x, y) = x' P y$$

.

The information on cross-dependencies and similarities among distinct features (dimensions) is here encoded in the off-diagonal matrix cells. This approach has become known recently as the so called "semantic" kernel approach. To make this approach valid a valid kernel the matrix P has to fulfill the property of being symmetric and even positive-semi-definite which is, however, easy to ensure.

Independent of the question which approach is chosen, two parameters influence the final result:

1. Will the representation of the initial features be crisp (as in standard concept instantiation) or will they be weighted (i.e. correspond to degrees of instantiation)?
2. Can the higher level features also occur as initial inputs or will the initial features be restricted to "leaf" nodes/concepts of the taxonomy/ontology ?
3. Will the integration of higher level features be crisp or weighted depending on the distance to the source feature ? For example in approach 2 this corresponds to the question, whether the matrix entries are restricted to the set $\{0, 1\}$ or whether they may take values from a real scale.

Looking in more detail at the third parameter, when choosing not to use crisp integration (which is, depending on the context likely to make too many items too similar) a whole set of variations for the weighting schemes exist based on research on semantic distance measures within taxonomies [Budanitsky, 2001, Ganesan et al., 2003].

5.2.3 Further Approaches

The above exposition has pointed to the directions that will form the main focused of investigation within the context of 3rd year bottom-up integration work in scenario 4. However, there are other related approaches.

An alternative direction stems from the field of *kernel methods for structured data*. It is common to distinguish two forms of structured data. On the one hand, the data items can be independent from each other but still be highly structured internally. On the other hand the data items can be are parts of a bigger structure (so called external structure) as for example individuals in a social network. Data that is described by means of ontological primitives can be seen as a natural form of both kinds of structured data. Kernel methods bring in a particular aspect, namely the formation of hypotheses by linear combination of positive-definite kernel functions. Kernel methods can be applied to different kinds of (structured) data by using an appropriately defined kernel functions which correspond to similarity functions in the sense explained above that are defined directly on that data [Shawe-Taylor and Cristianini, 2004, Gärtner, 2003]. There has been substantial work on kernels between internally structured instances such as graphs, trees, sequences,

and higher-order terms, as well as research on kernels between externally structured instances such as the vertices in graphs or hypergraphs.

Another direction is the use of background knowledge for *knowledge discovery with constraints*. The basic notion behind this field is that known constraints about the similarity of certain data items can be propagated back into the defined similarity function. The necessary background knowledge could for example be given within an ontological structure. The result of this approach is a skewed geometry of the feature space that makes use of the information given in the initial constraints. See e.g. [Klein et al., 2002] for a related approach in a clustering application.

5.2.4 Implementation Plan and Applications within SEKT

Initial experiments in the text classification domain using prototypical implementations and using the explicit feature expansion approach and showed good results. These experiments also showed that ontologies that have been automatically generated from textual data in a fully unsupervised process can show bring competitive improvements [Bloehdorn et al., 2006].

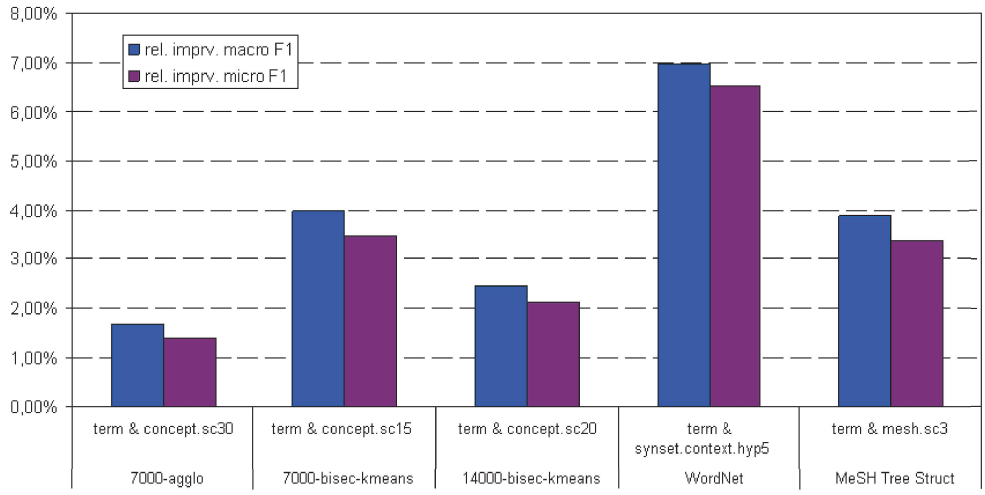

Figure 5.1: Relative Improvements of F_1 measure against the Bag-of-Words baseline in Text Classification Experiments on the Ohsumed Dataset using Feature Expansion based on manually engineered and learned ontologies. See [Bloehdorn et al., 2006] for details.

The integration of advanced similarity measures that make use of the background knowledge stored in ontologies into the TEXTGARDEN machine learning components will be the main integration activity within the 3rd year of the project.

WP3 will develop facilities for similarity calculations within the ontology within the

KAON 2 faremwork. This will be done namely between individual ontology and metadata entities, based on semantic distance measures as well as between complex (aggregated) descriptions based on ontology primitives.

Within scenario 4, these "semantic" similarities will be used as inputs for machine learning algorithms of any kind implemented within TEXTGARDEN. For example by a different plugins for the standard kernel module.

We aim at bringing advanced similarity measures into operation within at leas two case study scenarios. On the one hand, within the SEKT BT digital library case study (WP11), there is the need to improve both retrieval and recommender functionalities. Within the SEKT legal case study (WP10), advanced similarities can form an additional or alternative feature in the IUIRSERVICE II FAQ retrieval system [Casanovas et al., 2004, Ortiz et al., 2005].

Chapter 6

Conclusion

One of the main objectives of SEKT is to combining the three core technologies, namely Knowledge Discovery, Human Language Technology and Ontology/Metadata Management with the aim of achieving synergy effects by looking at similar tasks from different paradigms and enabling new functionalities.

Scenario	Scenario 1 Interoperability of Ontology Learning Tools	Scenario 2 Active Learning for TEXT2ONTO	Scenario 3 Meta-Learning for TEXT2ONTO	Scenario 4 Ontology based Similarities for ML algorithms
Components				
TEXTGARDEN-LIBRARY		•	•	•
TEXTGARDEN-ONTOGEN	•			
KAON-TEXT2ONTO	•	•	•	
KAON2 (OM & Reasoning)				•
Maturity				
Conceptual	high	high	low	advanced
Implementation	completed	prototype	not yet	not yet
Evaluation	ongoing	ongoing	not yet	not yet
Intended use				
WP09 (Siemens Study)	?	?	?	?
WP10 (Legal Case Study)	•	•	•	•
WP11 (BT DL Case Study)	•		•	•

Table 6.1: Integration Matrix

Parallel to the top-down integration via the gloal SIP platform, SEKT exploits a bottom-up integration strategy where the integration between the core technologies developed in SEKT is performed on a bilateral basis between the three core technical partners. This report is part of the bottom-up integration work performed in workpackage (WP) 6, specifically task 6.6 in which we have described the integration efforts between the Knowledge Discovery and Ontology/Metadata Management research fields by means of integration of the corresponding tool suites developed within SEKT, namely the TEXTGARDEN and KAON tools suites.

Results We have described a total of four different integration scenarios, two of which have been already achieved a high maturity while the remaining two scenarios are scheduled for the 3rd year of the project – table 6.1 summarizes the situation. At the current stage, a high level of integration has already been achieved especially in Scenario 1 - Integration of Ontology Learning Tools and scenario 2 - Active Learning for TEXT2ONTO.

Outlook Activities within months 25—36 will mainly focus on realizing scenarios 3 and 4 according to the descriptions. At the same time, work within scenarios 1 and 2 will be refined. While not planned explicitly, (new) scenarios beyond the upcoming scenarios 3 and 4 might be introduced and worked on.

Relation to SIP Integration The integration activities described in scenarios 1,2 and 3 are mainly related to software environments that are worked with at ontology engineering time and not at runtime of the SIP platform. As such, the question concerning integration via SIP does not apply here directly. If desirable or necessary, the interfaces between the components could just as well be wrapped within SIP pipelets, however, without bringing additional benefit to other components. The planned facilities for individual and composite similarity calculations from scenario 4 form a component that could be wrapped within a SIP pipelet for integration within the global SIP integration platform. Whether and how the actual integration will be pursued will largely depend on the intended functionalities of the overall platform and which further components are likely to make use of these components. Note however, that for the actual integration with the machine learning algorithms implemented in TEXTGARDEN where efficiency is crucial the additional overhead of SIP integration is unlikely to bring additional benefits.

Benefit to SEKT The benefit of the integration scenarios 1 — 3 lies in the combination of complementary approaches for ontology learning: on the one hand, the more structure driven-approaches of TEXT2ONTO and on the other hand the approaches of TEXTGARDEN which are inspired by large-scale machine learning. In all three cases, this is likely to bring additional flexibility and adaptivity with respect to the learning of ontologies within new domains and/or languages.

The activities pursued within scenario 4 aim at explicitly exploiting the framework for semantic descriptions of data items within machine learning paradigms. This will allow machine-learning oriented applications to take advantage of the rich semantic descriptions generated and used within SEKT. Specifically, the resulting similarity measures are a valuable ingredient for applications like for example the FAQ retrieval in the legal case study and recommendations in the BT digital library case study.

Overall Bottom-up Integration For the remaining pairs of core technologies, the parallel deliverables D6.6.2[1] and D.6.6.4[2] describe the Integration of Knowledge Discovery with Human Language Technology and Human Language Technology with Ontology Management respectively. These deliverables will be updated and extended in the upcoming deliverables D6.6.5 – D6.6.7 at the end of month 36, which will provide the final details on the bilateral efforts for exploring synergies of SEKT core technologies.

[1] D6.6.2 Integration of TextGarden with GATE
[2] D6.6.4 Integration of GATE with KAON

Bibliography

[Bloehdorn et al., 2006] Bloehdorn, S., Cimiano, P., and Hotho, A. (2006). Learning ontologies to improve text clustering and classification. In Spiliopoulou, M., Kruse, R., Nürnberger, A., Borgelt, C., and Gaul, W., editors, *From Data and Information Analysis to Knowledge Engineering: Proceedings of the 29th Annual Conference of the German Classification Society (GfKl 2005), Magdeburg, Germany, March 9-11, 2005*, volume 30 of *Studies in Classification, Data Analysis, and Knowledge Organization*. Springer. TO APPEAR.

[Bloehdorn et al., 2005] Bloehdorn, S., Haase, P., Sure, Y., Völker, J., Bevk, M., Bontcheva, K., and Roberts, I. (2005). Report on the integration of ML, HLT and OM. SEKT Deliverable D6.6.1, Institute AIFB, University of Karlsruhe.

[Bloehdorn and Hotho, 2004] Bloehdorn, S. and Hotho, A. (2004). Text classification by boosting weak learners based on terms and concepts. In *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM 2004), 1-4 November 2004, Brighton, UK*, pages 331–334. IEEE Computer Society.

[Budanitsky, 2001] Budanitsky, A. (2001). Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures.

[Casanovas et al., 2004] Casanovas, P., Poblet, M., Casellas, N., Vallbé, J.-J., Ramos, F., Benjamins, R., Blázquez, M., Rodrigo, L., Contreras, J., and Cruz, J. G. (2004). Legal scenario: Case study-intelligent integrated decision support for legal professionals. SEKT deliverable 10.2.1, Intelligent Software Components S.A. and Universitat Autonoma de Barcelona.

[Cimiano and Völker, 2005a] Cimiano, P. and Völker, J. (2005a). Text2onto - a framework for ontology learning and data-driven change discovery. In *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB'05)*.

[Cimiano and Völker, 2005b] Cimiano, P. and Völker, J. (2005b). 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.

[Fortuna et al., 2005a] Fortuna, B., Mladenić, D., and Grobelnik, M. (2005a). Ontology generation from scratch. SEKT deliverable 1.7.1, Jožef Stefan Institute.

[Fortuna et al., 2005b] Fortuna, B., Mladenić, D., and Grobelnik, M. (2005b). Visualization of text document corpus. *Informatica journal*, 29(4):497–502.

[Ganesan et al., 2003] Ganesan, P., Garcia-Molina, H., and Widom, J. (2003). Exploiting hierarchical domain structure to compute similarity. *ACM Trans. Inf. Syst.*, 21(1):64–93.

[Grobelnik and Mladenić, 1998] Grobelnik, M. and Mladenić, D. (1998). Learning machine : design and implementation. Jsi technical report (ijs-dp-7824), Jožef Stefan Institute.

[Gärtner, 2003] Gärtner, T. (2003). A survey of kernels for structured data. *SIGKDD Explor. Newsl.*, 5(1):49–58.

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

[Haase and Voelker, 2004] Haase, P. and Voelker, J. (2004). Requirements analysis for usage-driven and data-driven change discovery. SEKT informal deliverable 3.3.1.a, Institute AIFB, University of Karlsruhe.

[Hustadt et al., 2004a] Hustadt, U., Motik, B., and Sattler, U. (2004a). Reasoning for Description Logics around $\mathcal{SHIQ}$ n a Resolution Framework. Technical Report 3-8-04/04, FZI, Karlsruhe, Germany.
http://www.fzi.de/wim/publikationen.php?id=1172.

[Hustadt et al., 2004b] Hustadt, U., Motik, B., and Sattler, U. (2004b). Reducing $\mathcal{SHIQ}^-$ Description Logic to Disjunctive Datalog Programs. In Dubois, D., Welty, C., and Williams, M.-A., editors, *Proc. of the 9th Int. Conf. on Knowledge Representation and Reasoning (KR2004)*, pages 152–162, Menlo Park, California, USA. AAAI Press.

[Klein et al., 2002] Klein, D., Kamvar, S., and Manning, C. (2002). From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering.

[Maedche, 2002] Maedche, A. (2002). *Ontology Learning for the Semantic Web*. Kluwer Academics.

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

[Mladenić, 1996] Mladenić, D. (1996). Personal webwatcher: Implementation and design. Jsi technical report ijs-dp-7472, Jožef Stefan Institute.

[Motik et al., 2004] Motik, B., Sattler, U., and Studer, R. (2004). Query Answering for OWL-DL with Rules. In McIlraith, S. A., Plexousakis, D., and van Harmelen, F., editors, *Proc. of the 3rd Int. Semantic Web Conf. (ISWC 2004)*, volume 3298 of *Lecture Notes in Computer Science*, pages 549–563, Hiroshima, Japan. Springer.

[Novak et al., 2004a] Novak, B., Fortuna, B., Mladenić, D., and Grobelnik, M. (2004a). Collecting data for ontology generation. SEKT deliverable 1.1.1, Jožef Stefan Institute.

[Novak et al., 2004b] Novak, B., Grobelnik, M., and Mladenić, D. (2004b). Dealing with unlabelled data. SEKT deliverable 1.2.1, Jožef Stefan Institute.

[Novak et al., 2005] Novak, B., Mladenić, D., and Grobelnik, M. (2005). Text classification with active learning. In *Proc. of the 29th Annual Conference of the German Classification Society (GfKl 2005)*, pages 70–77.

[Ortiz et al., 2005] Ortiz, R. P., Cívico, M. B., Cino, J. C., Benjamins, R., Casanovas, P., and Casellas, N. (2005). Legal case study prototype informal deliverable. SEKT Informal Deliverable 10.3.1.1, Intelligent Software Components S.A. and Universitat Autonoma de Barcelona.

[Salton and McGill, 1983] Salton, G. and McGill, M. J. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, NY ,USA.

[Scott and Matwin, 1998] Scott, S. and Matwin, S. (1998). Text classification using WordNet hypernyms. In Harabagiu, S., editor, *Use of WordNet in Natural Language Processing Systems: Proceedings of the Conference*, pages 38–44. Association for Computational Linguistics, Somerset, New Jersey.

[Shawe-Taylor and Cristianini, 2004] Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. Cambridge University Press.

[Siolas and d'Alché Buc, 2000] Siolas, G. and d'Alché Buc, F. (2000). Support vector machines based on a semantic kernel for text categorization. In *IJCNN '00: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00)-Volume 5*, page 5205, Washington, DC, USA. IEEE Computer Society.

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

[Tong and Koller, 2000] Tong, S. and Koller, D. (2000). Support vector machine active learning with applications to text classification. In *Proceedings of 17th International Conference on Machine Learning (ICML 00)*, pages 999–1006.

Appendix A

Availability of Prototype Implementations

Scenario 1: Interoperability of Ontology Learning Tools

ONTOGEN is publicly available trough the website http://www.textmining.net/ and runs in the Windows operating system. ONTOGEN also needs .NET framework 2.0 which can be freely downloaded from internet.

ONTOGEN is publicly available trough the website `http://www.textmining.net/` and runs in the Windows operating system. The website also includes more detailed information on how to use the software, sample data for testing it and a demonstration in video format showing sample use of the software on the sample data. ONTOGEN also needs .NET framework 2.0 which can be freely downloaded from `http://msdn.microsoft.com/netframework/`.

Scenario 2: Active Ontology Learning

A stand-alone version of the active learning tool is available at `http://ontoware.org/projects/symi/`. The distribution contains the necessary Java libraries which need to reside on the classpath as well as the active learning command line tool from TEXTGARDEN. A start of the main class requires a call to `org.semanticweb.sekt.activeic.InstanceClassificationGUI` with the additional argument of a Java properties file which needs to specify the path to the windows executable for active learning through the property key "`command`".

In this mode, the component is run as a stand alone tool via the `main` method. However, the underlying methods are designed to interface with the TEXT2ONTO environment. A corresponding version of TEXT2ONTO will be prepared and used within year 3 evaluations.

Appendix B

Availability of TG and KAON software

TextGarden

TEXTGARDEN command line tools are publicly available trough the website `http://www.textmining.net/` and run on the Windows operating system. Scheduled releases for TEXTGARDEN will include different distribution formats, e.g. as library including adaptors for Matlab, Java and the like.

Text2Onto

TEXT2ONTO is available for downlaod at `http://ontoware.org/projects/text2onto/`. TEXT2ONTO is written entirely in Java and is thus platform independent. However, KAON2 requires JDK 1.5 and is not compatible with earlier Java versions.

KAON2

KAON2 is available as a precompiled binary distribution and is free of charge for research and academic purposes, see `http://kaon2.semanticweb.org/`. KAON2 is written entirely in Java and is thus platform independent. However, KAON2 requires JDK 1.5 and is not compatible with earlier Java versions.