D.6.6.1 Report on the integration of ML, HLT and OM

Stephan Bloehdorn, Peter Haase, York Sure and Johanna Voelker (University of Karlsruhe)

with contributions from
Matjaz Bevk (JSI), Kalina Bontcheva (USFD) and Ian Roberts (USFD)

Abstract.
EU-IST Integrated Project (IP) IST-2003-506826 SEKT
Deliverable D6.6.1 (WP6)

SEKT aims to integrate approaches from Machine Learning (ML), Human Language Technology (HLT) and Ontology and Metadata Management (OM). This deliverable is part of the SEKT bottom-up integration activities and lays the foundation for a number of deliverables (D6.6.x). It explores integration directions for these three research fields and reports on the ongoing and scheduled technical integration activities between the different software tool suites provided by the respective workpackages (WP1 – WP3).

Keyword list: SEKT Integration, Machine Learning, Text Mining, Human Language Technology, Ontology Learning Ontology Management

Document Id.	SEKT/2005/D6.6.1/v1.0
Project	SEKT EU-IST-2003-506826
Date	July 11th, 2005
Distribution	public

Copyright © 2005 Institute AIFB, University of Karlsruhe

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

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 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

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	23.05.05	Stephan Bloehdorn Peter Haase Johanna Völker	creation; input chapters 1, 2, 3
0.2	10.06.05	Stephan Bloehdorn Matjaz Bevk Johanna Völker	input for chapters 1, 3, 4
0.3	21.06.05	Matjaz Bevk	input for chapter 4
0.4	21.06.05	York Sure	included overview in chapter 1; overall polishing
0.5	24.06.05	Kalina Bontcheva Ian Roberts	input for chapters 3 and 4
0.5	24.06.05	Stephan Bloehdorn	extensions to all chapters, consistency check
0.6	08.07.05	Stephan Bloehdorn	changes according to reviewer comments
1.0	11.07.05	Stephan Bloehdorn York Sure	changes according to reviewer comments

Executive Summary

The vision of SEKT is to develop technologies for Next Generation Knowledge Management. In doing so, SEKT builds on the synergy and interplay between Machine Learning (ML), Human Language Technology (HLT) and Ontology and Metadata Management (OM), especially in the context of learning ontological and metadata structures.

This deliverable is the first in a row of deliverables concerned with the **bottom-up integration** activities of SEKT task 6.6, which aim at the integration of the aforementioned core technologies in light-weight proof-of-concept implementations and complements the top-down integration activities in task 6.5. Task 6.6 has been included after revision of results in task 6.5 at the end of year 1 of the SEKT project. It was felt necessary to better understand the low-level opportunities for integration — first to better understand the top-down integration activities, second to emphasize more the research needed to combine SEKT technologies. This deliverable is the first report from task 6.6 on

- possible integration directions for these three research fields,
- completed integration activities between the contributed software tool suites from the different fields,
- scheduled future integration activities.

At the current stage, good integration has been achieved especially between HLT (GATE) with OM (i.e. KAON-TEXT2ONTO) and at the same time ML (TEXTGARDEN tool suite) and HLT have also already shown integration potential. This deliverable anticipates the deliverables D6.6.2 – D6.6.7 that will report incrementally and in technical detail on the pairwise integration between the SEKT core technologies and their software suites.

Contents

Chapter 1

Introduction

SEKT aims at developing technologies for Next Generation Knowledge Management that exploit complementary research fields like Machine Learning (ML), Human Language Technology (HLT) and Ontology and Metadata Management (OM). Specifically, SEKT develops software to:

- semi-automatically learn ontologies and extract metadata,
- provide knowledge access,
- to perform middleware functionalities for global integration.

SEKT combines these core technologies to achieve synergy effects, whereby each technology is a key part of (semi-) automatic ontology learning (i.e. discovery of ontology primitives both on the schema and on the instance level) and maintenance.

This report is part of the bottom-up integration work performed in workpackage (WP) 6 on, 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 SEKT core technologies can be found among the 'Research & Development' activities: WP1 explores approaches for Ontology Generation from a machine-learning 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 infrasturcture further with focus on the KAON2 and TEXT2ONTO components. This report is part of the work performed in workpackage (WP) 6 on Integration and is naturally related with the technical workpackages and their interaction.

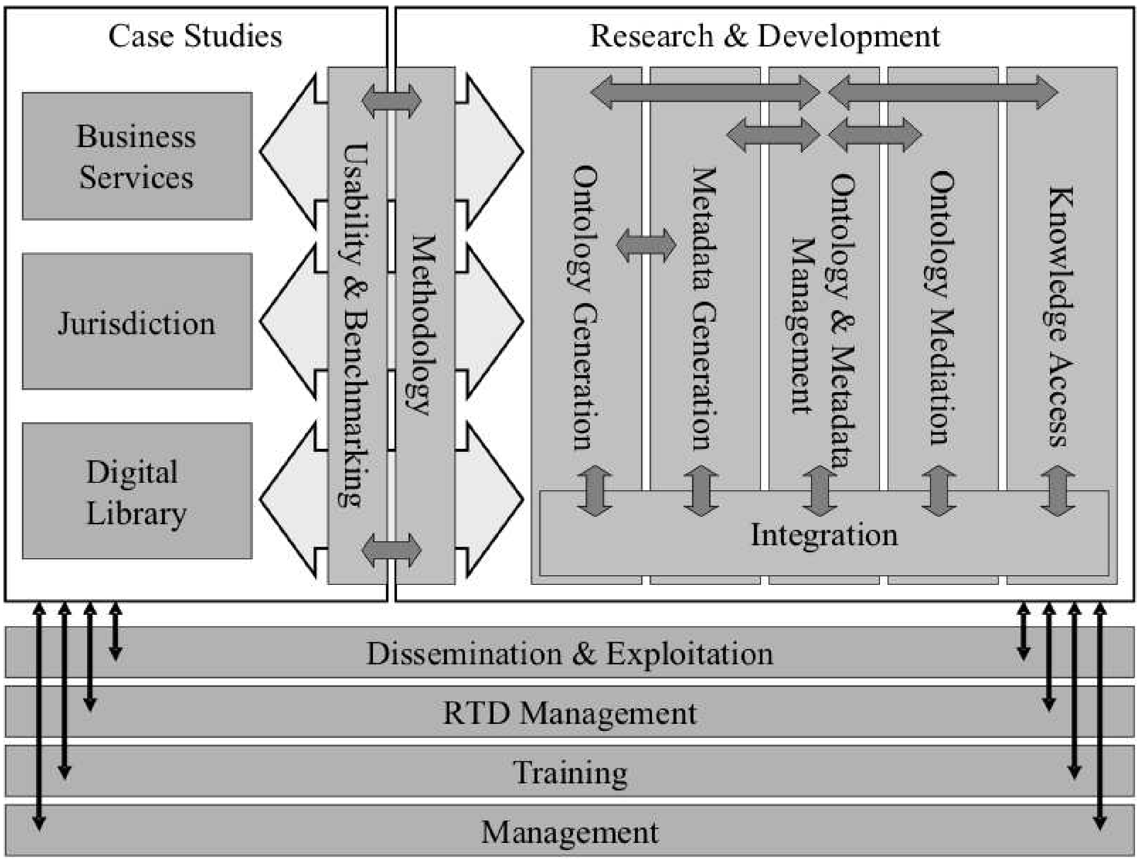

Figure 1.1: The SEKT Big Picture

1.2 Approaches to Integration in SEKT

Workpackage 6 aims at providing integration between the SEKT technical components in two ways. On the one hand it deals with a **top-down integration** approach 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.

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 so-called “pipelets”. Further information about the basic platform can be found e.g. in the SEKT deliverables D6.1 – D6.4. The deliverables D6.5.x describe the integration of components from the core workpackages into SIP, more specifically the integration of TEXTGARDEN, GATE and KAON (such as the OWL-DL inference engine KAON2) into SIP.

The top-down integration allows for bundling of integrated components in many different ways. This is particularly useful for bundling of SEKT technologies into enterprise solutions.

1.2.2 Bottom-Up Integration

In parallel, SEKT exploits a bottom-up integration where we integrate pairwise the core technologies developed in SEKT.

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. Bottom-up integration explores the synergies which the SEKT core technologies offer for (semi-)automatic creation and maintenance of ontologies, or briefly ontology learning. It was felt necessary to better understand the low-level opportunities for integration — first to better understand the top-down integration activities, second to emphasize more the research 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 machine learning, human language technologies and ontology management.
- Coordinate the efforts for integration of the core technical components delivered in WP1, WP2 and WP3.

This explores the synergies on a bilateral basis between workpackages 1–3. The bottom-up integration task 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.

This deliverable is the first of a series of bottom-up deliverables D6.6.x on integration from the second perspective, which will be complemented by the deliverables D6.6.2 – D6.6.7 that will report incrementally and in more technical detail on the bilateral integration efforts.

1.3 Outline

This deliverable is structured as follows: Chapter 2 provides a comprehensive definition of the (1) main ontology modelling primitives important for ontology learning tasks complemented by (2) an ontology learning meta-model that allows to express knowledge about ontology modelling primitives and learning tasks themselves, e.g. for modelling the software architecture and data flow of the ontology learning software developed within SEKT in upcoming deliverables. Chapter 3 shortly describes and reviews the software environments for ML, HLT and OM currently available within SEKT, namely the TEXTGARDEN, GATE and KAON (TEXT2ONTO and KAON2) tool suites. The current status of the global and mutual interaction of the individual tools is reported in chapter 4 together with the further scheduled integration between the individual tools. The report concludes in chapter 5.

Chapter 2

A Common Ontology Model

There are different notions in the literature and different research communities what ontologies are – or should be. Some 'definitions' of ontologies are discussed in [Guarino, 1997], the most prominent being *"An ontology is an explicit specification of a conceptualization"* [Gruber, 1995]. Here, a 'conceptualization' refers to an abstract model of some phenomenon in the world by identifying the relevant concept of that phenomenon while 'explicit' means that the types of concepts used and the constraints on their use are explicitly defined. This definition is often extended by three additional conditions: "An ontology is an explicit, *formal* specification of a *shared* conceptualization *of a domain of interest*". 'Formal' refers to the fact that the ontology should be machine readable, i.e. should adhere to a formal description which can be interpreted by machines in a fixed and predefined way. 'Shared' reflects the notion that an ontology captures consensual knowledge, that is, it is not private to some individual, but accepted as a group. The reference to 'a domain of interest' indicates that for domain ontologies one is not interested in modelling the whole world, but rather in modelling just the parts which are relevant to the task at hand. All kinds of related work on ontologies can be found in [Staab and Studer, 2004].

This section aims at providing two formal descriptions of ontologies which can be viewed from either perspective. The first formalization is described in section 2.1 and is inspired from a strict mathematical point of view. The second formalization, a meta-model for the ontology model, which focuses on comprehensibility for software engineers familiar with object oriented modelling is given in section 2.2.

The purpose of providing these formalizations is twofold: on the one hand, they provide a conceptual basis for the formalization of ontology learning and related tasks. By reducing the ontology model to a set of core primitives, it can be used more easily for extensions in different directions given a particular notion of the ontology learning tasks. An initial overview of ontology learning tasks as they are for example performed by TEXT2ONTO and partly by GATE is given in appendix A and will be extended in the future deliverables, especially in D6.6.4 and D6.6.7 on the integration between GATE and KAON (TEXT2ONTO). At the same time the formalizations, especially the meta-model can be used themselves for modelling the software architecture and data flow of the ontology learning software developed within SEKT in upcoming deliverables.

2.1 Formal Definitions of Ontology Modelling Primitives

This section aims at providing mathematically strict definitions of ontological modelling primitives which are present in almost all ontology definition languages. These definitions, which are largely based on our earlier formalizations of the notion of an ontology, e.g. in [Bozsak et al., 2002], are mainly meant to study structural aspects important for ontology learning tasks without committing ourselves too much to a specific ontology representation language[1]. The model relates to OWL-DLP, the fragment of the OWL ontology language used in SEKT, in the sense that all the modelling primitives listed here can be found in OWL-DLP. However, not all possible OWL constructs are reflected in the ontology model, it is restricted only to those that are likely to be discovered by ontology learning methods.

2.1.1 Core Ontology Modelling Primitives

Definition 2.1.1 (Ontology)**.** An *ontology* is a structure

$$\mathcal{O} := (C, \leq_C, R, \sigma, \leq_R, I, \iota_C, \iota_R)$$

consisting of

- three disjoint sets C, R and I whose elements are called *concept identifiers*, *relation identifiers* and *instance identifiers* respectively which we will also call *concepts*, *relations* and *instances* for convenience,
- a partial order $\leq_C$ on C, called *concept hierarchy* or *taxonomy*,
- a function $\sigma\colon R \to C^+$ called *signature*,
- a partial order $\leq_R$ on R, called *relation hierarchy*,
- a function $\iota_C\colon C \to \mathfrak{P}(I)$ called *concept instantiation*,
- a function $\iota_R\colon R \to \mathfrak{P}(I^+)$ called *relation instantiation* with $\iota_R(r) \subseteq \prod_{c\in\sigma(r)} \iota_C(c)$, for all $r \in R$.

[1]The definitions were mainly developed in long discussions in the Seminar of the knowledge management group at the institute AIFB at the University of Karlsruhe.

Some people argue that the term 'Ontology' should refer only to the formalization of the *intentional* aspects of a domain. To avoid confusion, we will refer to these *intentional* aspects as the *core ontology*. This notion is the same as the notion of *terminological knowledge* or *T-Box* in Description Logics. The complementary part is used for modelling extensional knowledge and is sometimes referred to as the *knowledge base*. This extensional part of an ontology contains assertions about instances of the concepts and relations analogously to *assertional knowledge (A-Box)* in Description Logics.

Definition 2.1.2 (Core Ontology and Knowledge Base)**.** Given an Ontology

$$\mathcal{O} = (C, \leq_C, R, \sigma, \leq_R, I, \iota_C, \iota_R)$$

we will call the structure

$$CO_{\mathcal{O}} := (C, \leq_C, R, \sigma, \leq_R)$$

the *core ontology* of the ontology $\mathcal{O}$ and the structure

$$KB_{\mathcal{O}} := (I, \iota_C, \iota_R)$$

the *knowledge base* of the ontology $\mathcal{O}$.

In ontologies, the terminological primitives, i.e. *concepts* and *relations* are arranged in hierarchical structures.

Definition 2.1.3 (Subconcepts and -relations)**.** If $c_1 \leq_C c_2$, for $c_1, c_2 \in C$, then c_1 is a *subconcept of* c_2, and c_2 is a *superconcept of* c_1. If $r_1 \leq_R r_2$, for $r_1, r_2 \in R$, then r_1 is a *subrelation of* r_2, and r_2 is a *superrelation of* r_1.

Sometimes interested in the first level further specialisation of concept or a relation. We can define these through the following definition.

Definition 2.1.4 (Direct Subconcepts and -relations)**.** If $c_1 \leq_C c_2$ and there is no $c_3 \in C$ with $c_3 \neq c_1$, $c3 \neq c_2$ and $c_1 \leq_C c_3 \leq_C c_2$, then c_1 is a *direct subconcept of* c_2, and c_2 is a *direct superconcept of* c_1. We note this by $c_1 \prec_C c_2$. *Direct superrelations* and *direct subrelations* are defined analogously.

In practice, relations are typically binary. For those relations, we define their *domain* and their *range*.

Definition 2.1.5 (Domain and Range)**.** For a relation $r \in R$ with $|\sigma(r)| = 2$, we define its *domain* and its range by $\mathrm{dom}(r) := \pi_1(\sigma(r))$ and $\mathrm{range}(r) := \pi_2(\sigma(r))$.

For two relations $r_1, r_2 \in R$, $r_1 \leq_R r_2$ implies $|\sigma(r_1)| = |\sigma(r_2)|$ and $\pi_i(\sigma(r_1)) \leq_C \pi_i(\sigma(r_2))$, for each $1 \leq i \leq |\sigma(r_1)|$ where $\pi_i(\sigma(.))$ denotes the i-th argument specified by $\sigma(.)$.

When a knowledge base is given, we can derive the extensions of the concepts and relations of the ontology, based on the concept instantiation and the relation instantiation.

Definition 2.1.6 (Concept and Relation 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 $c' <_C c$.

The *extension* $[\![r]\!] \subseteq I^+$ *of a relation* $r \in R$ is recursively defined by the following rules:

- $[\![r]\!] \leftarrow \iota_R(r)$
- $[\![r]\!] \leftarrow [\![r]\!] \cup [\![r']\!]$, for $r' <_R r$.

2.1.2 Lexical Ontology Modelling Extensions

According to the international standard ISO 704 [ISO 704, 2000], we provide names for the concepts (and relations). Taking terminology from semiotics, we will often call them 'sign' instead of 'name' to allow for more generality.

Definition 2.1.7 (Lexicon for an Ontology)**.** A *lexicon* for an ontology $\mathcal{O}$ is a structure

$$Lex := (S_C, S_R, S_I, Ref_C, Ref_R, Ref_I)$$

consisting of

- three sets $S_C \supseteq C$, $S_R \supseteq R$ and $S_I \supseteq I$ whose elements are called *signs for concepts*, *signs for relations* and *signs for instances* and
- a relation $Ref_C \subseteq S_C \times C$ called *lexical reference for concepts*, where $(c, c) \in Ref_C$ holds for all $c \in C$.
- a relation $Ref_R \subseteq S_R \times R$ called *lexical reference for relations*, where $(r, r) \in Ref_R$ holds for all $r \in R$.
- a relation $Ref_I \subseteq S_I \times I$ called *lexical reference for instances*, where $(i, i) \in Ref_I$ holds for all $i \in I$.

Based on Ref_C, we define, for $s \in S_C$,

$$Ref_C(s) := \{c \in C \mid (s, c) \in Ref_C\}$$

and, for $c \in C$,

$$Ref_C^{-1}(c) := \{s \in S \mid (s, c) \in Ref_C\} \ .$$

Ref_R, Ref_R^{-1}, Ref_I and Ref_I^{-1} are defined analogously.

Definition 2.1.8 (Ontology with Lexicon)**.** An *ontology with lexicon* is a tuple

$$(\mathcal{O}, Lex)$$

where $\mathcal{O}$ is an ontology and Lex is a lexicon for $\mathcal{O}$.

2.2 Metamodel for Learned Ontologies

Complementary to the ontology model presented in the previous section, we now present a metamodel for the ontology model introduced in the previous section. A metamodel is used to define the terms that a model is to be defined in. For the definition of the metamodel, we follow the standard of the OMG for metamodeling, i.e. MOF2 (Meta Object Facility). MOF2 defines an abstract language and framework for specifying, constructing and managing technology neutral metamodels that can be exploited for describing ontology models (see for example [Djuric et al., 2003] for a related approach). The main benefits of using a metamodel based on MOF2 are the following:

- It increases the comprehensibility for engineers familiar with object oriented modeling, i.e. the proposed metamodel enables approaching ontologies in the way that is closer to software engineering practitioners.
- It provides a visual (UML-based) notation for models, complementary to the more abstract, mathematical model.
- If grounded in (and thus compatible with) MOF2 allows the reuse of existing tools for UML based modelling, transformation, integration, validation, etc. of the models.

2.2.1 Core Ontology and Knowledge Base Metamodels

Figure 2.1 shows the metamodel and an exemplary corresponding model instantiation of the ontology core and knowledge base.

We will briefly discuss the correspondences between the ontology model and the metamodel. We begin with the core ontology:

- Concepts directly represent the set of concepts C of the ontology model. The concept hierarchy $\leq_C$ is represented as a relation subConceptOf between concepts.
- Relations the set of relations R of the ontology model. The relation hierarchy $\leq_R$ is represented as a relation subRelationOfbetween relations. The relations domain and range capture the signature σ. Note that the metamodel follows the ontology model presented in the previous section in capturing binary relations only.

The knowledge base is captured by:

- Instances represent the set of instances I of the ontology model. They *instantiate* concepts as defined by the concept instantiation function ι_C.

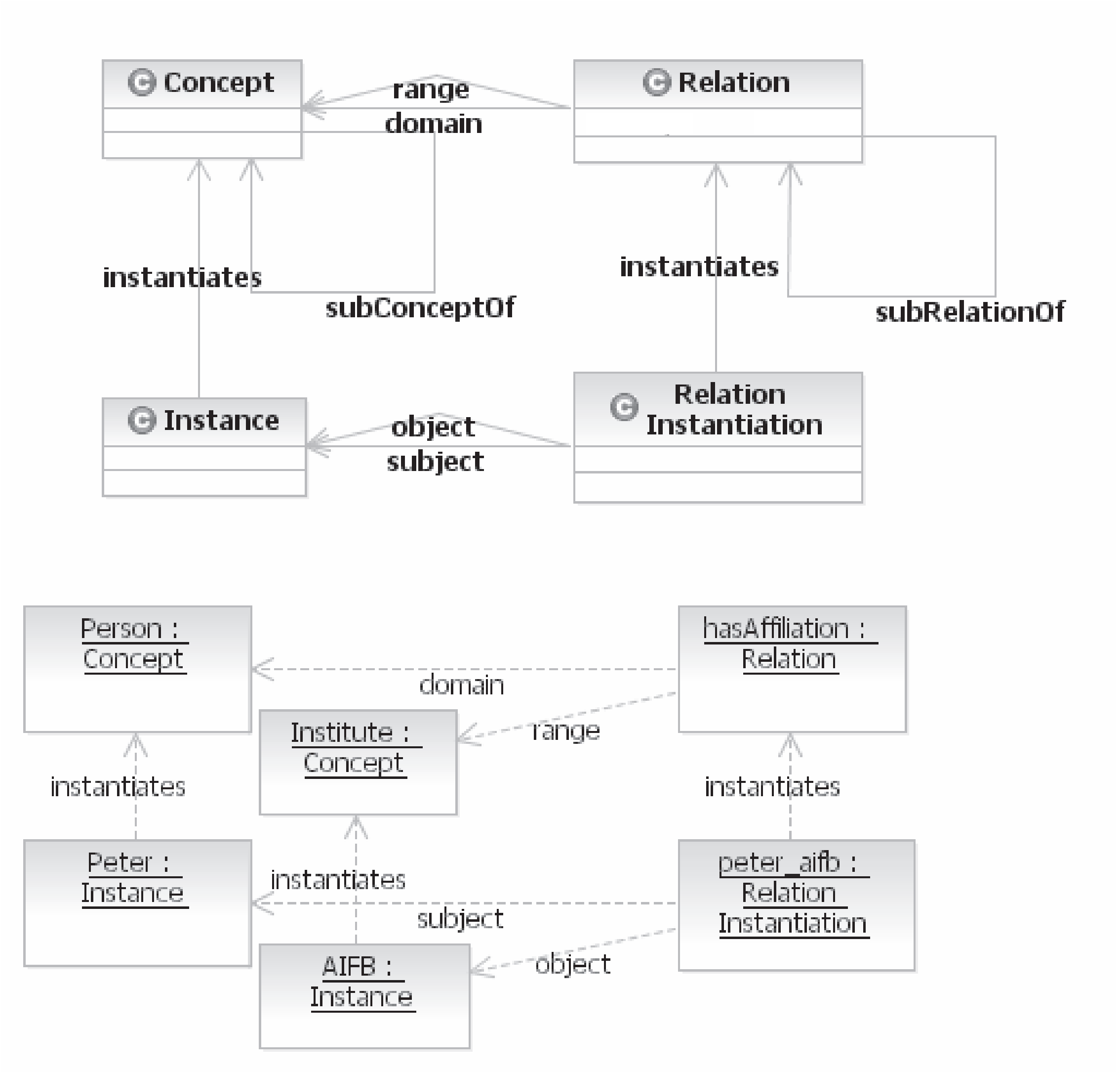

Figure 2.1: Core Ontology and Knowledge Base

- Relation Instantiations are modelled as first class objects to represent function ι_R, i.e. the extent of relation. For binary relations, the subject (the first argument - i.e. an instance from the domain) and object (the second argument - i.e. an instance from the range) of the relation are model with dedicated relations.

The lower part of the figure shows a concrete model (instance) of the metamodel.

2.2.2 Extensions: Additional Ontology Metamodels

The ontology meta model presented in the previous section provides a very generic, abstract view on ontologies. This view is sufficient for a large number of activities that perform ontology learning tasks, i.e. the (semi-)automatic discovery of ontology structures form textual or other data. Appendix A lists a number of tasks targeted by ontology learning tools. For certain cases, however, this view does not meet the notion of certain other ontology learning tasks as worked on in SEKT (e.g. in workpackage 1), we here present a further metamodel as a possible extension for the example of learning topic hierarchies. We aim at collecting similar extensions in the future and reporting them in the upcoming deliverables D6.6.x .

Topic Hierarchies Topic hierarchies such as ACM [2] or DMOZ(ODP) [3] are often used to organize collections of documents in a structured, mainly hierachical manner. These topic hierarchies are often considered light-weight ontologies. However, the semantics of a topic hierarchy is typically not adequately captured by existing ontology models, especially with respect to inheritence of the specialication relation. In general there are many ways to represent topic hierarchies[4]. Each of the presented alternatives have advantages and disadvantages - there is no agreed "perfect" approach. We therefore propose to model topic hierarchies using a dedicated metamodel as shown in Figure 2.2.

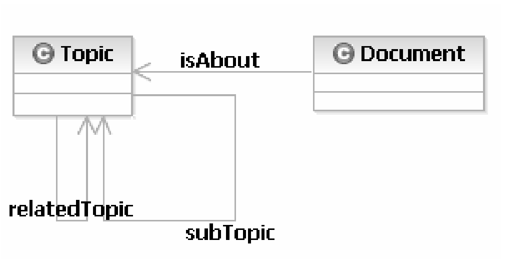

Figure 2.2: Document Classification

This allows us to abstract from a specific representation of in the ontology model[5]. The meta model consists of the following classes:

- Topics represent the nodes in a topic hierarchy. They are associated with other topics via specialication relations (subTopic) and references to related topics.
- Documents are the entities to be classified against the topic hierarchy. The relationship with topics is established via the isAbout relation.

[2] `http://www.acm.org/class/1998/`
[3] `http://dmoz.org/`
[4] `http://www.w3.org/TR/swbp-classes-as-values/`
[5] We can then of course later define various mappings from this meta model to that of the ontology meta model.

Chapter 3

Description of Core SEKT Tools

To make this deliverable self-contained, we shortly describe the set of tools developed most prominently within in workpackages 1–3 from the ML, HLT and OM research fields respectively. This chapter is meant as a reference for the next chapter that will refer back to these software tools.

3.1 Text Garden

TEXTGARDEN tools[1] is a set of software components developed mainly within workpackage 1 (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.

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

- Pre-processing of Text Documents in various formats,
- Indexing of Text Documents and efficient representation in the Bag-of-Words File format,
- Document classification including automatic model generation based on various Machine Learning Algorithms including Support Vector Machines, Logistic Regression, Winnow and more,
- Processing of unlabelled Data and Active Learning (c.f. D1.2.1 [Novak et al., 2004b]),

[1]see `http://kt.ijs.si/Dunja/textgarden/`

- Document clustering using various clustering algorithms including k-Means and hierarchical clustering,
- Document Visualization,
- Dealing with Web documents,
- Crawling the Web (c.f. D1.1.1 [Novak et al., 2004a])

and many other.

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 or similar utility TEXTGARDEN can already be run on Linux/Unix platforms. The release of a C++ library for easier integration is planned, too.

3.2 General Architecture for Text Engineering (GATE)

GATE [Cunningham et al., 2002] is one of the most widely used human language processing systems in the world[2] and has been built over the past eight years in the Sheffield NLP group. It is a tool for: scientists performing experiments that involve processing human language, companies developing applications with language processing components and teachers/students of courses about language/language computation. GATE can been used for many language processing projects, in particular for Information Extraction (IE) in many languages. In the context of SEKT, GATE is further developed and extended mainly in workpackage 2 (Metadata Generation) with a strong focus on Ontology-Aware Adaptive IE in order to make the automatic and semi-automatic adaptation of GATE to ontologies possible: change the ontology and change the extraction system. Also, recent advances in mixedinitiative learning and unsupervised language data models will be deployed to increase the adaptivity of metadata generation tools like GATE to evolving end-user information needs. Besides this, the focus of the development work is on Controlled Language IE and Deep Structure Analysis.

Functionalities GATE supports the full lifecycle of language processing components, from corpus collection and annotation through system evaluation.

Linguistic data in GATE is associated with language resources such as documents and corpora and is encoded in the form of annotations. GATE supports a variety of formats including XML, RTF, HTML, SGML, email and plain text. In all cases, when a

[2]GATE is freely available for download from `http://gate.ac.uk`.

document is created/opened in GATE, the format is analyzed and converted into a single unified model of annotation. The annotation format is a modified form of the TIPSTER format [Grishman, 1997] which has been made largely compatible with the Atlas format [Bird et al., 2000], and uses 'stand-off markup' [Thompson and McKelvie, 1997]. The annotations associated with each document are a structure central to GATE, because they encode the language data read and produced by each processing module. Each annotation has a start and an end offset and a set of features associated with it. Each feature has a "name" and an associated "value", which holds the descriptive or analytical information such as Part-of-speech and sense tags, syntactic analysis, named entities identification and co-reference information etc.

JAPE, Java Annotation Patterns Engine, is part of the GATE system. It is an engine based on regular expression pattern/action rules over annotations. JAPE is a version of CPSL – (Common Pattern Specification Language). This engine executes the JAPE grammar phases - each phase consists of a set of pattern/action rules. The left-hand-side (LHS) of the rule represents an annotation pattern and the right-hand-side (RHS) describes the action to be taken when pattern found in the document. JAPE executes these rules in a sequential manner and applies the RHS action to generate new annotations over the matched regular expression pattern. Rule prioritisation (if activated) prevents multiple assignments of annotations to the same text string. JAPE is being used in TEXT2ONTO in the development of ontology learning patterns (see details in the next chapter).

Technical Details GATE comprises three elements: an architecture describing how language processing systems are made up of components, a framework, i.e. a class library/SDK and a graphical development environment built on the framework. GATE is written in Java and tested on Linux, Windows and Solaris.

3.3 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 [3] which will be described in the following.

[3]KAON2 is a successor to the original KAON project (referred to as KAON1). The main difference to KAON1 is the supported ontology language: KAON1 used a proprietary extension of RDFS, whereas KAON2 is based on OWL-DL. KAON2 is a completely new system, and is not backward-compatible with KAON1. At the same time, TEXT2ONTO is a complete re-design and re-engineering of KAON-TEXTTOONTO which was part of the original KAON project.

3.3.1 Text2Onto

TEXT2ONTO [Cimiano and Völker, 2005] 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 (see section 2.2) within a so called *Probabilistic Ontology Model (POM)*. TEXT2ONTO thus remains independent of a concrete target language 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[4] 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.

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

3.3.2 KAON2

KAON2[5] 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].

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,
- 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 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.

[5]KAON2 is available as a precompiled binary distribution and is free of charge for research and academic purposes, see `http://kaon2.semanticweb.org/`

Chapter 4

Report on current Tool integration

Task 6.6 which deals with bottom-up integration has started at month 13, i.e. at the beginning of the second year. Detailed descriptions of the integration activities performed are scheduled for reporting at the end of the second year by the respective partners in the following deliverables (D6.6.2 – D6.6.7). These will provide details on the bilateral efforts for exploring synergies of SEKT core technologies. However, since the efforts are already ongoing, this section is meant as a intermediate status report and we briefly present an overview what is already performed and what is planned for the next months.

Remark: *To allow to put the bilateral integration activities into the overall context of the project, especially with respect to the top-down integration activities, we append to the reports on the integration of the individual tools also a short summary of the status of the Integration with the SEKT Integration Plattform (SIP).*

4.1 Satus of Text Garden Integration Activities

4.1.1 Integration with GATE

ML and HLT offer a certain potential for integration. TEXTGARDEN tools currently rely on statistical heuristics when identifying chunks of natural language in text documents. GATE provides means for deeper linguistic analysis like lemmatization or part-of-speech information. However, efficiency considerations restrict these to certain cases.

Current State of the Integration Currently, no GATE functionalities have been integrated for use by the TEXTGARDEN components. However, a part of TEXTGARDEN for working with SVM algorithm has been integrated into GATE. TEXTGARDEN components are command line utilities and have been integrated as external program calls from

GATE, which is written in Java. The following utilities of TEXTGARDEN have been integrated: Txt2Bow, BowTrainBinSVM and BowClassify. See below for further details.

Scheduled Integration A conceptual framework and details for the Integration of GATE functionalities into TEXTGARDEN will be presented in the upcoming SEKT deliverable 6.6.2 in month 24.

4.1.2 Integration with KAON

Just as linguistic background knowledge, ontologies have shown potential for improving text mining tasks by enhancing the classical bag-of-words representation known from information retrieval [Salton and McGill, 1983] with additional features on a conceptual level, resulting in better results of the respective text mining tasks [Bloehdorn and Hotho, 2004, Hotho et al., 2003]. Similarly, ontologies provide a certain potential for improving algorithms by improving similarity calculations among feature vectors [Siolas and d'Alché Buc, 2000].

Current State of the Intergration Currently, no TEXT2ONTO or KAON2 functionalities have been integrated for use by the TEXTGARDEN components, conceptual considerations sketch an integration based on the ideas above.

Scheduled Integration Ideas for using conceptual features derived from ontologies for Machine Learning, especially Text Mining tasks will be pursued. A detailed conceptual framework for these ideas and details for the Integration between TEXT2ONTO/KAON2 functionalities into TEXTGARDEN will be presented in the upcoming SEKT deliverable 6.6.3 in month 24.

4.1.3 Integration with SIP

Current State of the Intergration WP 1 has started with the integration of TEXTGARDEN into SIP. To keep integration simple and effective, TEXTGARDEN tools are integrated as SIP pipelets. Here, each integrated pipelet is a java wrapper which calls a corresponding command line utility from TEXTGARDEN toolset. Two types of communication between sequential utilities in the pipeline are supported. The first way is using blackboard object, which is native to the SIP platform, the second is the communication via the file system which is native to TextGarden. The following TEXTGARDEN utilities have already been integrated into SIP:

Txt2Bow Transforms various raw text formats, such as Text-Base, Transactions-File, Compact-Documents-File and also some standard datasets into the file in Bag-Of-

Words format (.Bow). Required input is a set of text documents, either as plain text or HTML files in the file system or in one of the many other input formats handled by TEXTGARDEN; output is a bag-of-words file.

BowKMeans Performs K-Means clustering on the document in Bag-Of-Words format (.Bow) and outputs the clustering of documents in different formats, such as text file or XML file.

GetCentroidWords This is not an actual TextGarden utility but needed for reading the results of BowKMeans clustering. Reads the centroid words from a clustering created by BowKMeans pipelet.

BowTrainBinSVM Learns a model via training a binary-class Support Vector Machine on the set of input documents provided in the Bag-Of-Words format, outputs the corresponding model in a native TEXTGARDEN format.

BowClassify Classifies input documents provided in the Bag-Of-Words format using the provided model.

Scheduled Integration Integration with SIP has so far covered only a few areas which are supported by TEXTGARDEN. These are pre-processing (Text2Bow), clustering (BowKMeans), learning (BowTrainBinSVM) and classification (BowClassify). In the near future we plan to extend coverage in these areas and the additional tacled by TEXTGARDEN as well. The following list contains a list of TEXTGARDEN utilities to be integrated into SEKT platform:

Document clustering:

BowHKMeans : Performs hierarchical K-Means clustering on the document in Bag-Of-Words format (.Bow) and outputs the clustering of documents in different formats, such as text file or XML file.

Learning a model for classification:

BowTrainLogReg Learns a model using logistic regression on the set of input documents provided in the Bag-Of-Words format.

BowTrainWinnow Learns a model via training a Winnow on the set of input documents provided in the Bag-Of-Words format.

BowTrainPerceptron Learns a model via training a Perceptron on the set of input documents provided in the Bag-Of-Words format.

Feature extraction:

Bow2SemSpace Creates semantic-space representation of documents based on Latent Semantic Indexing.

Visualization:

Bow2VizGraph Creates graph representation of input documents provided in the Bag-Of-Words format and outputs .xml file.

Bow2VisTile Creates tiling representation of input documents provided in the Bag-Of-Words format and outputs .xml file.

Bow2VizMap Provides visualization of documents as a 2-D map based on the semantic-space representation previously generated using Bow2SemSpace.

4.2 Satus of GATE Integration Activities

4.2.1 Integration with TextGarden

At several points, linguistic analysis can benefit from machine learning algorithms. Consequently, GATE could benefit from using TEXTGARDEN components for machine learning tasks.

Current State of the Intergration As discussed above, TEXTGARDEN consists of a number of machine learning algorithms implemented as stand-alone executables in C++. The integration into GATE is achieved by implementing Java wrappers which write out and read in the files expected by TEXTGARDEN modules and execute the TEXTGARDEN components. The Text Garden wrappers provide access to the Text Garden implementations through the Machine Learning PR in GATE. The wrappers implements all the temporary file handing internally, the user only interacts with GATE.

A typical workflow in which GATE and TEXTGARDEN cooperate looks like this: GATE writes out to the file a description of a learning problem, which could be for example feature vectors for words. Further on, GATE executes TEXTGARDEN utilities Txt2Bow and BowTrainBinSVM to obtain a model, which can later be used to classify new instances using BowClassify utility. Consequently, the TextGarden executables Txt2Bow, BowTrainBinSVM and BowClassifyBow must be on your path to allow the wrapper to find them.

Currently, Text Garden SVM only supports binary classification, i.e. the CLASS attribute must be boolean. The other attributes can be boolean, numeric or nominal, or any combination of these. If an attribute is nominal, each value of that attribute maps to a separate SVM feature. Each of these SVM features will be given the value 1 when the nominal attribute has the corresponding value, and will be omitted otherwise. If the value of the nominal is not specified in the configuration file or there is no value for an instance, then no feature will be added. Text Garden models are not updateable, and so are created and trained the first time a classification is attempted. However, the Wrapper supports the batch classification mode of the machine learning processing resource. If <BATCH-MODE-CLASSIFICATION/> is specified in the <ENGINE> part of the configuration file, then all the instances for a document will be passed to the wrapper at one time, rather than them being passed one at a time. Using this option will result in a great improvement in efficiency in most circumstances.

There are two options for the wrapper:

- TRAINER-OPTIONS: These are the same options as those passed to BowTrainBinSVM on the command line.
- CLASSIFIER-OPTIONS: These are the same options as those passed to BowClassifyBow on the command line.

Scheduled Integration Beside the current integration status, additional TEXTGARDEN components will be analyzed for potential integration into GATE. Details of the integration of TEXTGARDEN functionalities into GATE will be presented in the upcoming SEKT deliverable 6.6.2 in month 24.

4.2.2 Integration with KAON

An obvious direction for integration is to provide facilities for GATE to access KAON2 ontologies, be alerted of ontology evolution events, etc. This might, for example, be used in the context of the Ontological Gazetteer functionalities in GATE for attaching instances of concepts in texts to ontologies, i.e. performing the task of learning concept instantiations as defined in appendix A.

Current State of the Integration Currently, no TEXT2ONTO or KAON2 functionalities are exploited by GATE. Initial discussions direct the attention to approaches like the ones mentioned above. The other way round, TEXT2ONTO integrate a number of GATE functionalities as discussed below.

Scheduled Integration While GATE modules can now be used in TEXT2ONTO and thus provide input to the ontology, future efforts are planned in order to enable GATE modules to access the ontology in KAON2 as discussed above.The first integrated prototype is planned for month 24 and will be reported on in SEKT deliverable 6.6.4.

4.2.3 Integration with SIP

Current State of the Integration Provided with GATE is a set of reusable processing resources for common NLP tasks. (None of them are definitive, and the user can replace and/or extend them as necessary.) These are packaged together to form ANNIE, A Nearly-New IE system, but can also be used individually or coupled together with new modules in order to create new applications. For example, many other NLP tasks might require a sentence splitter and POS tagger, but would not necessarily require resources more specific to IE tasks such as a named entity transducer. The system is in use for a variety of IE and other tasks, sometimes in combination with other sets of application-specific modules.

ANNIE consists of the following main processing resources: tokeniser, sentence splitter, POS tagger, gazetteer, finite state transducer (based on JAPE), and orthomatcher. The resources communicate via GATE's annotation API, which is a directed graph of arcs bearing arbitrary feature/value data, and nodes rooting this data into document content (in this case text).

The *tokeniser* splits text into simple tokens, such as numbers, punctuation, symbols, and words of different types (e.g. with an initial capital, all upper case, etc.). The aim is to limit the work of the tokeniser to maximise efficiency, and enable greater flexibility by placing the burden of analysis on the grammars. This means that the tokeniser does not need to be modified for different applications or text types.

The *sentence splitter* is a cascade of finite-state transducers which segments the text into sentences. This module is required for the tagger. Both the splitter and tagger are domain- and application-independent.

The *tagger* is a modified version of the Brill tagger, which produces a part-of-speech tag as an annotation on each word or symbol. Neither the splitter nor the tagger are a mandatory part of the NE system, but the annotations they produce can be used by the grammar (described below), in order to increase its power and coverage.

The *gazetteer* consists of lists such as cities, organisations, days of the week, etc. It not only consists of entities, but also of names of useful *indicators*, such as typical company designators (e.g. 'Ltd.'), titles, etc. The gazetteer lists are compiled into finite state machines, which can match text tokens.

The *semantic tagger* consists of hand-crafted rules for identifying named entities, e.g., persons, organisations, locations, dates, money amounts, etc. The rules are written in JAPE (Java Annotations Pattern Engine) [Cunningham et al., 2005] and describe patterns to match and annotations to be created as a result. Patterns are specified by describing a specific text string, or annotations previously created by modules such as the tokeniser, gazetteer, or document format analysis. Rule prioritisation prevents multiple assignment of annotations to the same text string.

The *orthomatcher* is another optional module for the IE system. Its primary objective is to perform co-reference, or entity tracking, by recognising relations between entities. It also has a secondary role in improving named entity recognition by assigning annotations to previously unclassified names, based on relations with existing entities.

Scheduled Integration In order to enable easy use of ANNIE in SEKT, it is being integrated by Empolis in SIP as a pipelet, in a manner similar to TEXTGARDEN (see above). Further details of this integration appear in D6.5.2.

4.3 Satus of KAON Integration Activities

4.3.1 Integration with TextGarden

Plans for using TEXTGARDEN together with KAON can be grouped in two directions:

- store TEXTGARDEN results as instance data with respect to a suitable ontology (e.g. PROTON or suitable PROTON extensions to make them available for further processing.
- use TEXTGARDEN to pre-process text documents for further ontology learning activities by means of TEXT2ONTO.

Current State of the Integration Currently, no TEXTGARDEN functionalities are exploited by TEXT2ONTO or KAON2. Technical Discussions on TEXTGARDEN outputs in OWL/RDF have started. Also, a conceptual framework for using TEXTGARDEN in the context of TEXT2ONTO aim at providing document preprocessing functionalities by means of TEXTGARDEN components have begun. It is expected that clustering documents or classifying documents against a set of thematic classes will allow TEXT2ONTO to reduce the noise introduced by documents that do not match a certain domain. These preprocessing components will call TEXTGARDEN components from TEXT2ONTO.

Scheduled Integration Details on these ongoing activities will be reported in SEKT deliverable D.6.6.3.

4.3.2 Integration with GATE

Plans for improving ontology access for GATE through KAON2 have been sketched above. Within KOAN the use of GATE is primarily for ontology learning activities.

Current State of the Integration Many existing ontology learning environments focus either on pure machine learning techniques [Bisson et al., 2000] or rely on linguistic analysis [Buitelaar et al., 2003, Velardi et al., 2005] 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. For this purpose, TEXT2ONTO has been integrated with GATE. As a result, TEXT2ONTO is 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.

Specifically, (1) the ANNIE components of GATE and some other GATE PRs (e.g., lemmatisers, morphological analyzers, stemmers) are integrated in TEXT2ONTO, in order to support ontology learning and (2) the JAPE engine of GATE which provides finite state transduction over annotations based on regular expressions is used within TEXT2ONTO.

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 For TEXT2ONTO we developed JAPE patterns for both shallow parsing and the identification of modelling primitives, i.e. concepts, instances and different types of relations (c.f. [Hearst, 1992]).

Scheduled Integration Since obviously, 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. TEXT2ONTO plans to make an extensive use of any new or upcoming multilingual facilities in GATE, especially the German and Spanish language tools developed as part of SEKT, deliverable D1.4.1. The German, French and Spanish modules provided or under development within GATE are as follows:

- multilingual tokeniser
- German/French/Spanish sentence splitters
- German/French/Spanish part-of-speech (POS) taggers
- English/French/German gazetteers

- German/French semantic tagger
- Spanish keyword annotator
- multilingual coreferencer

Since TEXT2ONTO should provide full support for all of these languages in future releases, it will integrate these components, and TEXT2ONTO will include appropriate patterns for Spanish and German.

4.3.3 Integration with SIP

Current State of the Intergration Initial versions of the KAON2/ SIP interfaces have been designed and are scheduled for integration in the close future.

Scheduled Integration Based on the ongoing interface definitions, the integration of KAON2 into SIP will be completed in the near future. By means of these interfaces, also TEXT2ONTO results can be read from KAON2.

Chapter 5

Conclusion

Automatically learning ontological structures and corresponding metadata is one of the key challenges of Next Generation Knowledge Management. As a result, it is one of the major topics to be covered in SEKT and aims to integrate approaches to the problem from the diverse areas Machine Learning, Human Language Technology and Ontology and Metadata Management. In this deliverable, we have presented an initial status report on the ongoing work in the bottom-up integration between the software components developed in SEKT workpackages 1–3.

We started with a formal specification of a simple ontology model that captures most of the primitives important for ontology learning tasks and can be used and extended in future deliverables. In the remainder, we have shortly presented the tools developed within SEKT and have described initial ideas, current integration status and scheduled integration activities for these tools. Due to the ongoing nature of these activities these presentations are meant to be seen as intermediate results.

At the current stage, a high level of integration has already been achieved especially between HLT (GATE) and OM (i.e. KAON-TEXT2ONTO). At the same time ML (TEXTGARDEN tool suite) and HLT have also already shown integration potential. For the remaining tools, this deliverable has sketched the current integration discussions. Detailed descriptions of the integration activities performed within the second year are scheduled for reporting in month 24 by the respective partners in the upcoming bilateral deliverables (D6.6.2 – D6.6.7). These will provide details on the bilateral efforts for exploring synergies of SEKT core technologies.

Bibliography

[Bird et al., 2000] Bird, S., Day, D., Garofolo, J., Henderson, J., Laprun, C., and Liberman, M. (2000). ATLAS: A flexible and extensible architecture for linguistic annotation. In *Proceedings of the Second International Conference on Language Resources and Evaluation*, Athens.

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

[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.

[Bozsak et al., 2002] Bozsak, E., Ehrig, M., Handschuh, S., Hotho, A., Maedche, A., Motik, B., Oberle, D., Schmitz, C., Staab, S., Stojanovic, L., Stojanovic, N., Studer, R., Stumme, G., Sure, Y., Tane, J., Volz, R., and Zacharias, V. (2002). KAON – towards a large scale semantic web. In Bauknecht, K., Tjoa, A. M., and Quirchmayr, G., editors, *Proceedings of the Third International Conference on E-Commerce and Web Technologies (EC-Web 2002)*, volume 2455 of *LNCS*, pages 304–313, Aix-en-Provence, France. Springer.

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

[Cimiano and Völker, 2005] Cimiano, P. and Völker, J. (2005). 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'2005).* to appear.

[Cunningham et al., 2002] Cunningham, H., Maynard, D., Bontcheva, K., and Tablan, V. (2002). GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Annual Meeting of the ACL.*

[Cunningham et al., 2005] Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., Ursu, C., Dimitrov, M., Dowman, M., Aswani, N., and Roberts, I. (2005). *Developing Language Processing Components with GATE Version 3 (a User Guide)*. `http://gate.ac.uk/`.

[Djuric et al., 2003] Djuric, D., Gaševic, D., and Devedžic, V. (2003). A mda-based approach to the ontology definition metamodel. In *Proceedings of the 4th International Workshop on Computational Intelligence and Information Technologies, Nis, Serbia and Montenegro*, pages 51–54.

[Grishman, 1997] Grishman, R. (1997). TIPSTER Architecture Design Document Version 2.3. Technical report, DARPA. `http://www.itl.nist.gov/div894/-894.02/related_projects/tipster/`.

[Gruber, 1995] Gruber, T. R. (1995). Towards principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43(5/6):907–928.

[Guarino, 1997] Guarino, N. (1997). Understanding, building and using ontologies. *International Journal of Human and Computer Studies*, 46(2/3):293–310.

[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.

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

[Hotho et al., 2003] Hotho, A., Staab, S., and Stumme, G. (2003). Ontologies improve text document clustering. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), 19-42 November 2003, Melbourne, Florida, USA*, pages 541–544. IEEE Computer Society.

[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.

[ISO 704, 2000] ISO 704 (2000). Terminology work — principles and methods. Technical report, International Organization of Standardization (ISO).

[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).

[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.

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

[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.

[Staab and Studer, 2004] Staab, S. and Studer, R., editors (2004). *Handbook on Ontologies in Information Systems.* International Handbooks on Information Systems. Springer.

[Thompson and McKelvie, 1997] Thompson, H. and McKelvie, D. (1997). Hyperlink semantics for standoff markup of read-only documents. In *Proceedings of SGML Europe'97*, Barcelona.

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

Appendix A

Core Ontology Learning Tasks

In this section, we identify a number of ontology learning tasks which are expressed in terms of the definitions given in the last chapter.

Concept Extraction aims at discovering concepts $c_1, c_2, c_3 \ldots$ relevant for the domain modeled by the ontology that should be added to the set C of concepts in $\mathcal{O}$.

In terms of the Metamodel this corresponds to learning instantiations $[\![Concept]\!]$ of concept Concept. The representation / definition of each concept c depends on the specific ontology learning methods used. But no matter how it looks like it should allow to distinguish between different concepts and to determine whether two concepts are equal.

Concept Labelling The concept labelling task is closely related to the concept extraction task and often, they will be regarded or solved as one task. In terms of the aforementioned ontology model, concept labelling refers to learning signs $s \in S_C$ for concepts $c \in C$. Typically, this will be the main label of the concept but it may also include synonymous signs. In this case the task is not only to find a valid label but also to discover which of the synonymous signs for a certain concept should be used as the label.

Learning Concept Hierarchies Given a set C of concepts in $\mathcal{O}$, this task is to infer taxonomic relationships $\leq_C$ in the ontology. Specifically, this task typically aims at identifying all pairs $c_1, c_2 \in C$ for which $c_1 \prec_C c_2$, i.e. a direct subconcept relation holds to infer the general taxonomic relationships $\leq_C$ from this.

In terms of the metamodel, this corresponds to learning subConceptOf relationships between instantiations of Concept.

Instance Extraction Similar to the concept extraction task, the instance extraction task aims at the identification of potential instances $[\![c]\!]$ of concepts $c \in C$. In terms of the metamodel this means learning instantiations $[\![Instance]\!]$ of concept Instance.

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

Relation Extraction Identification of relations $r \in R$. In the metamodel: Learning instantiations $[\![Relation]\!]$ of concept $Relation$.

Relation Labelling Correspondingly, relation labelling refers to learning signs S_R for relations $r \in R$.

Learning Relation Hierarchies Similar to the learning of concept hierarchies this tasks aims at learning the relation hierarchy $\leq_R$ for the ontology. Specifically, this task aims at identifying all pairs $r_1, r_2 \in R$ for which $r_1 \prec_R r_2$, i.e. a direct subrelation exists.

Learning Relation Instantiations Learning instantiations $[\![r]\!] \leftarrow \iota_R(r)$ for $[\![r]\!] \subseteq \mathfrak{P}(I)$ and $r \in R$. Meta Model: Learning (i) instantiations $[\![RelationInstantiation]\!]$ of concept $RelationInstantiation$ and (ii) $instantiates$ relationships between instances of $RelationInstantiation$ and $Relation$.